

**IOAN TOMESCU
ADRIAN LEU**

**MATEMATICĂ
APLICATĂ
ÎN
TEHNICA
DE CALCUL**

X



IOAN TOMESCU
ADRIAN LEU

MATEMATICĂ APLICATĂ ÎN TEHNICA DE CALCUL

Clasa a X-a

Manual pentru licee cu profil de
matematică și de matematică-
fizică



Editura Didactică și Pedagogică,
București

I. ALGEBRĂ BOOLEANĂ

I.1. Noțiunea de algebră booleană

I.1.1. Lattice

În cadrul aritmeticii și algebrei sînt studiate diferite operații cu numere. În logica matematică s-au introdus operații ca: disjuncția, conjuncția sau negația propozițiilor. În cazul mulțimii părților unei mulțimi s-au întîlnit operațiile de reuniune, intersecție și complementară. Operațiile amintite se deosebesc mult între ele, în primul rînd pentru că se efectuează asupra unor elemente aparținînd unor mulțimi diferite.

Este cunoscut că operațiile pot avea sau nu anumite proprietăți. Operații diferite, care se efectuează asupra unor elemente distincte, pot avea aceleași proprietăți. De exemplu, adunarea, înmulțirea, reuniunea, intersecția, conjuncția și disjuncția sînt operații comutative și asociative.

Asemănarea unor operații din punctul de vedere al proprietăților pe care le posedă a dus la ideea de a nu se ține seama de semnificația concretă a acestor operații sau a elementelor mulțimii pe care sînt definite, ci de a se lua în considerare numai proprietățile lor comune. Trecerea de la operații determinate la operații nedeterminate este asemănătoare cu trecerea de la aritmetică la algebră, adică de la efectuarea unor operații cu numere cunoscute, la operații cu numere necunoscute.

Dacă pe o mulțime sînt definite una sau mai multe operații, care au anumite proprietăți, vom spune că mulțimea a fost dotată cu o structură algebrică. Mulțimea însăși, împreună cu operațiile definite, se numește uneori algebră. Desigur, denumirea sugerează unele asemănări cu algebra obișnuită, dar trebuie subliniate cele două sensuri ale cuvîntului algebră și anume ramură a matematicii, respectiv mulțime dotată cu o anumită structură.

Algebra booleană este un caz particular de algebră, operațiile definite trebuind să satisfacă anumite relații bine determinate. Pentru a ajunge la definiția structurii de algebră booleană se va prezenta mai întîi o structură mai simplă, denumită lattice.

Definiție. Se numește *lattice* o mulțime nevidă M înzestrată cu două operații binare, notate \sqcap și \sqcup , astfel încît pentru oricare elemente $a, b, c \in M$ să fie valabile următoarele proprietăți:

$$1. a \sqcap b = b \sqcap a$$

$$1'. a \sqcup b = b \sqcup a \text{ (comutativitate)}$$

$$2. (a \sqcap b) \sqcap c = a \sqcap (b \sqcap c)$$

$$2'. (a \sqcup b) \sqcup c = a \sqcup (b \sqcup c) \\ \text{(asociativitate)}$$

$$3. a \sqcap (a \sqcup b) = a$$

$$3'. a \sqcup (a \sqcap b) = a \text{ (absorbție)}$$

O lattice se notează $\langle M, \sqcap, \sqcup \rangle$. Dacă nu există posibilitatea unei confuzii, notația poate fi simplificată, renunțînd la specificarea celor două operații.

Exemple

1. Mulțimea părților unei mulțimi M notată $\mathcal{P}(M)$, înzestrată cu operațiile de intersecție și reuniune formează o latice. Proprietățile 1, 2 și 3 pentru intersecție, respectiv 1', 2' și 3' pentru reuniune sînt cunoscute din teoria mulțimilor. Această latice poate fi notată $\langle \mathcal{P}(M), \cap, \cup \rangle$.
2. Să considerăm mulțimea divizorilor pozitivi ai unui număr natural k . Această mulțime poate fi înzestrată cu două operații binare, care corespund calculului celui mai mare divizor comun și calculului celui mai mic multiplu comun. Astfel $a \sqcap b = \text{cmmdc}(a, b)$, iar $a \sqcup b = \text{cmmmc}(a, b)$. Proprietățile 1, 2, 1' și 2' sînt cunoscute, rezultînd direct din definițiile cmmdc și cmmmc . Proprietatea 3 se poate demonstra astfel: deoarece $a \sqcup b$ este cmmmc al numerelor a și b , rezultă că $a \sqcup b$ este un multiplu al numărului a ; în acest caz $a \sqcap (a \sqcup b)$ adică cmmdc al numerelor a și $a \sqcup b$, nu poate fi decît a . Analog se demonstrează și proprietatea 3'. Această mulțime formează o latice, care poate fi notată $\langle V, \text{cmmdc}, \text{cmmmc} \rangle$, unde $V = \{x \mid x \text{ este un divizor pozitiv al lui } k\}$.

Observație. Analizînd cele șase proprietăți din definiția dată unei latice se remarcă unele similitudini. Astfel, înlocuind în oricare dintre proprietăți o operație prin cealaltă (și reciproc) se obține tot o proprietate din definiție. Așadar, se poate enunța *principiul dualității* pentru latice, care are formularea:

Dacă într-o propoziție adevărată din teoria laticelor se înlocuiește o operație prin cealaltă (și reciproc) se obține de asemenea o propoziție adevărată.

Această a doua propoziție este numită *propoziția duală* corespunzătoare primei propoziții.

Principiul dualității permite să se evite demonstrarea unei propoziții atunci cînd propoziția sa duală este demonstrată.

Propoziție. În orice latice L , pentru orice element $a \in L$ sînt adevărate relațiile $a \sqcup a = a$ și $a \sqcap a = a$.

Demonstrație. Din proprietatea 3 rezultă $a = a \sqcap (a \sqcup b)$, astfel că

$$a \sqcup a = a \sqcup (a \sqcap (a \sqcup b)).$$

Notînd $a \sqcup b$ cu c , deoarece $c \in L$ rezultă conform proprietății 3' $a \sqcup (a \sqcap c) = a$, astfel că $a \sqcup a = a$.

Analog $a \sqcap a = a \sqcap (a \sqcup (a \sqcap b)) = a \sqcap (a \sqcup d) = a$, unde s-a notat $a \sqcap b$ cu d .

Observații

1. În orice latice se poate introduce o relație de ordine, notată \sqsubseteq , astfel: $a \sqsubseteq b \Leftrightarrow a \sqcup b = b$. Se verifică ușor că sînt adevărate cele trei proprietăți ale unei relații de ordine și anume:

1) reflexivitate: $a \sqsubseteq a$, deoarece $a \sqcup a = a$;

2) tranzitivitate: $a \sqsubseteq b$ și $b \sqsubseteq c \Rightarrow a \sqsubseteq c$, deoarece din $a \sqcup b = b$ și $b \sqcup c = c$ rezultă $a \sqcup c = a \sqcup (b \sqcup c) = (a \sqcup b) \sqcup c = b \sqcup c = c$;

3) antisimetrie: $a \sqsubseteq b$ și $b \sqsubseteq a \Rightarrow a = b$, deoarece din $a \sqcup b = b$ și $b \sqcup a = a$ rezultă $a = b \sqcup a = a \sqcup b = b$.

2. Se poate arăta că relația de ordine poate fi definită și prin:

$$a \sqsubseteq b \Leftrightarrow a \sqcap b = a.$$

Într-adevăr, dacă $a \sqcup b = b$, atunci

$$a \sqcap b = a \sqcap (a \sqcup b) = a \text{ și, similar,}$$

$$\text{dacă } a \sqcap b = a, \text{ atunci}$$

$$a \sqcup b = (a \sqcap b) \sqcup b = b \sqcup (b \sqcap a) = b.$$

În cazul celor două exemple de latice, relația de ordine corespunde relației de incluziune a două mulțimi și respectiv relației de divizibilitate a două numere.

Definiție. Simbolul p se numește **prim element** în laticea L dacă pentru orice $x \in L$ este adevărată relația $p \sqsubseteq x$. Analog, u se numește **ultim element** în laticea L dacă pentru orice $x \in L$ este adevărată relația $x \sqsubseteq u$.

Propoziție. În orice latice finită există un prim element p și un ultim element u în raport cu relația de ordine \sqsubseteq .

Demonstrație. Fie $L = \{a_1, a_2, \dots, a_n\}$. Să arătăm că

$$p = a_1 \sqcap a_2 \sqcap \dots \sqcap a_n \text{ (parantezele au fost omise datorită asociativității).}$$

Fie $x \in L$ un element oarecare al laticei L . Presupunând că acest element se află în poziția i , putem considera $x = a_i$.

Astfel

$$\begin{aligned} p \sqcap x &= (a_1 \sqcap a_2 \sqcap \dots \sqcap a_i \sqcap \dots \sqcap a_n) \sqcap a_i = \\ &= a_1 \sqcap a_2 \sqcap \dots \sqcap a_n = p, \end{aligned}$$

adică $p \sqsubseteq x$.

Similar, $u = a_1 \sqcup a_2 \sqcup \dots \sqcup a_n$, deoarece

$$\begin{aligned} u \sqcup x &= (a_1 \sqcup a_2 \sqcup \dots \sqcup a_i \sqcup \dots \sqcup a_n) \sqcup a_i = \\ &= a_1 \sqcup a_2 \sqcup \dots \sqcup a_n = u, \end{aligned}$$

adică $x \sqsubseteq u$.

Definiție. Dacă într-o latice L , pentru oricare elemente $a, b, c \in L$ sînt satisfăcute relațiile

$$a \sqcup (b \sqcap c) = (a \sqcup b) \sqcap (a \sqcup c),$$

$$a \sqcap (b \sqcup c) = (a \sqcap b) \sqcup (a \sqcap c),$$

laticea se numește **latice distributivă**.

Exemple

1. Laticea $\langle \mathcal{P}(M), \cap, \cup \rangle$ este distributivă deoarece operațiile de reuniune și intersecție din teoria mulțimilor sînt distributive una față de cealaltă.
2. Laticea $\langle \{x \mid x \text{ este divizor pozitiv al lui } k\}, \text{cmmdc}, \text{cmmmc} \rangle$ este distributivă deoarece pentru oricare trei numere $a, b, c \in \mathbb{N}$ sînt adevărate relațiile $\text{cmmmc}(a, \text{cmmdc}(b, c)) = \text{cmmdc}(\text{cmmmc}(a, b), \text{cmmmc}(a, c))$ și $\text{cmmdc}(a, \text{cmmmc}(b, c)) = \text{cmmmc}(\text{cmmdc}(a, b), \text{cmmdc}(a, c))$.
3. Fie mulțimea poligoanelor convexe din plan pentru care definim operațiile \sqcap și \sqcup astfel:
a) $P_1 \sqcap P_2 = P_1 \cap P_2$, adică intersecția din teoria mulțimilor aplicată celor două mulțimi de puncte ale suprafețelor poligonale P_1 și P_2 .

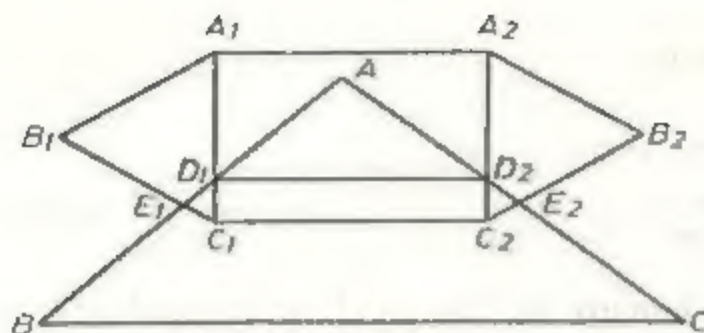


Fig. 1.1

b) $P_1 \sqcup P_2$ este cel mai mic poligon convex care conține pe $P_1 \cup P_2$.

Aceste operații verifică relațiile din definiția unei lățe (vezi ex. 6). Să arătăm că această lățe nu este distributivă.

Fie $P = \triangle ABC$, $P_1 = \triangle A_1B_1C_1$ și $P_2 = \triangle A_2B_2C_2$ (vezi fig. 1.1).

$$P_1 \sqcup P_2 = A_1B_1C_1C_2B_2A_2,$$

$$P \cap (P_1 \sqcup P_2) = \triangle A_1C_1C_2E_2,$$

$$P \cap P_1 = \triangle C_1D_1E_1,$$

$$P \cap P_2 = \triangle C_2D_2E_2,$$

$$(P \cap P_1) \sqcup (P \cap P_2) = D_1E_1C_1C_2E_2D_2.$$

Se observă că $\triangle A_1C_1C_2E_2 \neq \triangle D_1E_1C_1C_2E_2D_2$.

Observații

1. Cele două relații din definiția unei lățe distributive sînt duale una în raport cu cealaltă, astfel că principiul dualității este valabil și în lățele distributive.
2. Se poate arăta că într-o lățe cele două condiții de distributivitate sînt echivalente (vezi exercițiul 7).

1.1.2. Lățe distributive și complementate

Definiție. O lățe distributivă $\langle L, \cap, \sqcup \rangle$ se numește **complementată** dacă sînt îndeplinite următoarele condiții:

1. există un element neutru u pentru operația \cap (denumit **element universal**), care satisface relația

$$a \cap u = a \text{ pentru orice } a \in L;$$

2. există un element neutru n pentru operația \sqcup (denumit **element nul**), care satisface relația

$$a \sqcup n = a \text{ pentru orice } a \in L;$$

3. pentru orice element $a \in L$ există un element complementar $\bar{a} \in L$, care satisface relațiile

$$a \cap \bar{a} = n \text{ și}$$

$$a \sqcup \bar{a} = u.$$

Exemple

1. Lățe $\langle \mathcal{P}(M), \cap, \cup \rangle$ este o lățe distributivă complementată, deoarece mulțimea M este element neutru pentru intersecție, mulțimea vidă \emptyset este element neutru pentru reuniune, iar pentru orice mulțime $A \subseteq M$, mulțimea $\mathcal{C}_M A = M - A$ este o mulțime complementară a mulțimii A .
2. Dacă numărul k este compus doar din factori primi la puterea 1, atunci lățe $\langle \{x \mid x \text{ este divizor pozitiv al lui } k\}, \text{cmmdc}, \text{cmmmc} \rangle$ este distributivă și complementată.

Numărul k este element neutru pentru prima operație, deoarece cîmde $(x, k) = x$ pentru orice x divisor pozitiv al lui k , iar numărul 1 este element neutru pentru a doua operație deoarece cîmde $(x, 1) = x$ pentru orice x .

Dacă pentru un element x considerăm numărul $\frac{k}{x}$, atunci cîmde $\left(x, \frac{k}{x}\right) = 1$ și cîmde

$\left(x, \frac{k}{x}\right) = k$, deoarece aceste numere nu au divizori comuni. Rezultă, deci, că numărul $\frac{k}{x}$ este complementar numărului x .

Dacă numărul k are factori primi la puteri mai mari decît 1, atunci laticea divizorilor nu este complementată. Considerînd $k = 60 = 2^2 \cdot 3 \cdot 5$, nu se poate găsi pentru numărul 10 un număr care să satisfacă condițiile de complementaritate. Prima condiție este satisfăcută de numerele 1 și 3 care nu satisfac însă și cea de-a doua condiție (cîmde $(10, 1) = 10 \neq 60$ și cîmde $(10, 3) = 30 \neq 60$).

Propoziție. Într-o latice distributivă și complementată fiecare element are un singur element complementar.

Demonstrație. Fie o latice distributivă și complementată L și să presupunem, prin reducere la absurd, că există $a \in L$, care are două elemente complementare, \bar{a} și \bar{a}' , distincte ($\bar{a} \neq \bar{a}'$).

$$\begin{aligned}\bar{a} &= \bar{a} \sqcup n \quad (n \text{ este element neutru pentru } \sqcup) \\ &= \bar{a} \sqcup (a \sqcap \bar{a}') \quad (\bar{a}' \text{ este complement al lui } a) \\ &= (\bar{a} \sqcup a) \sqcap (\bar{a} \sqcup \bar{a}') \quad (\text{distributivitatea operației } \sqcup \text{ față de } \sqcap) \\ &= u \sqcap (\bar{a} \sqcup \bar{a}') \quad (\bar{a} \text{ este complement al lui } a) \\ &= \bar{a} \sqcup \bar{a}' \quad (u \text{ este element neutru pentru } \sqcap).\end{aligned}$$

Analog,

$$\begin{aligned}\bar{a}' &= \bar{a}' \sqcup n = \bar{a}' \sqcup (a \sqcap \bar{a}) = (\bar{a}' \sqcup a) \sqcap (\bar{a}' \sqcup \bar{a}) = \\ &= u \sqcap (\bar{a}' \sqcup \bar{a}) = \bar{a}' \sqcup \bar{a}.\end{aligned}$$

Deci

$$\begin{aligned}\bar{a} &= \bar{a} \sqcup \bar{a}' \\ &= \bar{a}' \sqcup \bar{a} \quad (\text{comutativitatea operației } \sqcup) \\ &= \bar{a}',\end{aligned}$$

în contradicție cu ipoteza $\bar{a} \neq \bar{a}'$.

Definiție. O latice distributivă și complementată se numește **algebră booleană**.

O algebră booleană se va nota $\langle A, \sqcap, \sqcup, \bar{}, u, n \rangle$.

Observații

1. Putem considera, în virtutea teoremei precedente, că într-o algebră booleană se poate defini o operație unară, care constă din luarea complementului unui element, astfel că într-o algebră booleană se poate spune că există trei operații \sqcap , \sqcup și $\bar{}$. Într-o algebră booleană cele trei operații sînt notate, de obicei, „ \cdot ”, „ $+$ ” și „ $-$ ”, iar cele două elemente neutre sînt notate, respectiv, prin 1 și 0.
2. Condițiile din definiția algebrei booleene nu sînt independente. S-a arătat, de exemplu, că pentru o latice distributivă este suficientă numai una dintre cele două condiții din definiție. Se poate arăta că pentru a defini o algebră booleană este suficient ca pentru oricare elemente a, b, c să fie adevărate relațiile

$$ab = ba,$$

$$a(b + c) = ab + ac,$$

$$a1 = a$$

$$a\bar{a} = 0,$$

$$a + b = b + a;$$

$$a + bc = (a + b)(a + c);$$

$$a + 0 = a;$$

$$a + \bar{a} = 1.$$

Celelalte relații, adică

$$(ab)c = a(bc), (a + b) + c = a + (b + c);$$

$$a(a + b) = a, a + ab = a$$

putând fi deduse din primele.

3. Elementul nul este prim element, iar elementul universal este ultim element (vezi exercițiul 8).
4. S-a arătat, de către Stone, că orice algebră booleană finită $\langle A, \cdot, +, \bar{}, 1, 0 \rangle$ poate fi pusă în corespondență bijectivă cu mulțimea părților unei mulțimi finite M , care este de asemenea, o algebră booleană $\langle \mathcal{P}(M), \cap, \cup, \complement_M, M, \emptyset \rangle$. În acest fel rezultă că numărul de elemente dintr-o algebră booleană finită A este 2^n , unde n reprezintă numărul de elemente ale mulțimii finite M asociate algebrei booleene A .

În continuare, se vor prezenta două exemple de algebre booleene utilizate în tehnica de calcul.

Algebra B_2 .

Algebra B_2 este algebra booleană formată din două elemente. Aceste elemente vor fi chiar elementele neutre ale celor două operații: 0 și 1. Putem considera că această algebră este cea mai simplă algebră booleană. Cele trei operații ale ei se definesc astfel:

$$0 + 0 = 0, \quad 0 \cdot 0 = 0, \quad \overline{0} = 1,$$

$$0 + 1 = 1, \quad 0 \cdot 1 = 0, \quad \overline{1} = 0.$$

$$1 + 0 = 1, \quad 1 \cdot 0 = 0,$$

$$1 + 1 = 1, \quad 1 \cdot 1 = 1.$$

Pentru a arăta că relațiile din definiția algebrei booleene sînt satisfăcute, se verifică egalitățile înlocuind elementele a, b, c cu 0, și 1, în toate combinațiile posibile. De exemplu, absorbția poate fi verificată ca în tabelul I.1.

a	b	$a + b$	$a(a + b)$	ab	$a + ab$
(1)	(2)	(3)	(4)	(5)	(6)
0	0	0	0	0	0
0	1	1	0	0	0
1	0	1	1	0	1
1	1	1	1	1	1

Tabelul I.1

Identitatea coloanelor (1), (4) și (6) demonstrează valabilitatea relațiilor $a = a(a + b) = a + ab$.

Se observă că operația \cdot coincide cu înmulțirea aritmetică, însă operația $+$ este diferită de adunarea aritmetică, deoarece $1 + 1 = 1$. Găsim, de exemplu, $a + b = \max(a, b)$, pentru orice $a, b \in B_2$.

Această algebră a avut o deosebită importanță în evoluția calculatoarelor. O însemnată contribuție teoretică și practică în domeniul algebrelor booleene și al aplicațiilor lor a adus matematicianul român Grigore C. Moisil (1906—1973).



Fig. 1.2

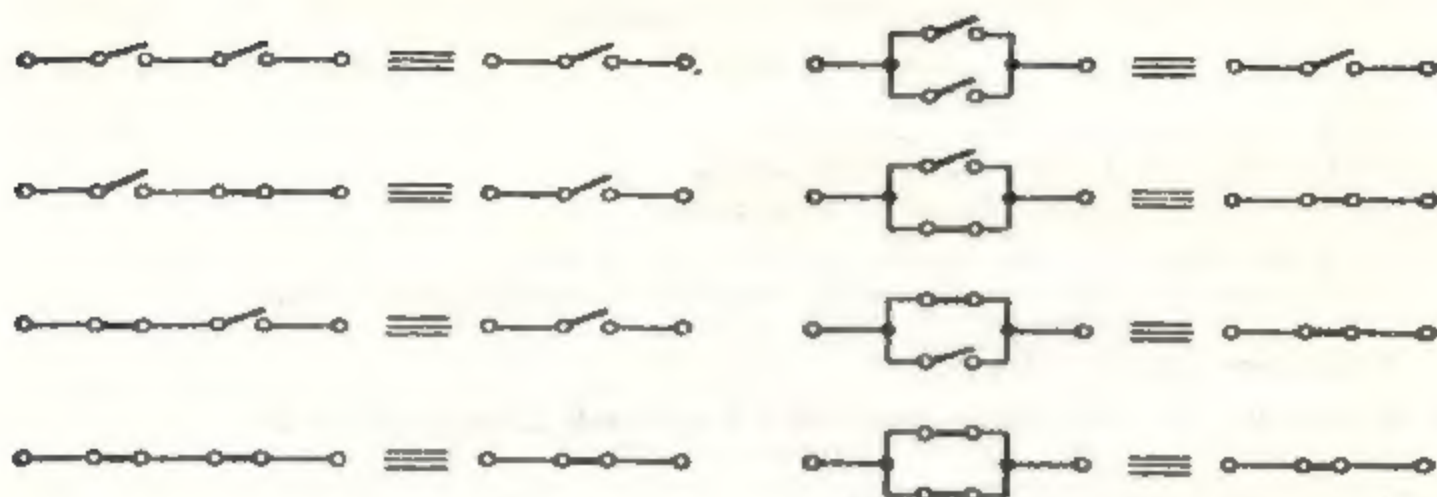


Fig. 1.3

Elementele algebrei B_2 pot fi interpretate ca reprezentind cei doi dipoli (rețea cu două borne) din figura 1.2.

Operațiile \cdot și $+$ sînt interpretate ca legarea dipolilor în serie, și respectiv, în paralel. Doi dipoli sînt considerați echivalenți dacă amîndoi, fie permit trecerea curentului electric, fie nu o permit.

Se poate observa, în figura 1.3, că această interpretare este în concordanță cu definiția celor două operații.

Operația de complementare se interpretează analog și corespunde la schimbarea stării dipolului. Astfel, dacă dipolul permite trecerea curentului, dipolul complementar nu o va permite (și invers).

Algebra propozițiilor

În cadrul mulțimii propozițiilor se pot defini operațiile de disjuncție, conjuncție și negație. Dacă se consideră egale propozițiile echivalente, atunci relațiile de comutativitate, asociativitate și distributivitate corespunzătoare acestor operații sînt satisfăcute. Elementul universal este format de clasa tautologiilor (propoziții întotdeauna adevărate), iar elementul nul este reprezentat de clasa contradicțiilor (propoziții întotdeauna false). Se verifică ușor că aceste elemente neutre satisfac relațiile din definiția unei algebre booleene. Deci, mulțimea claselor de echivalență a propozițiilor formează o algebră booleană, fiecare clasă conținînd propoziții echivalente logic.

1.1.3. Exerciții

1. Să se arate că relația

$$x \sqcup (y \sqcap z) = (x \sqcup y) \sqcap z$$

este verificată în orice latice dacă:

- a) $x = z$,
- b) $x = p$,
- c) $z = u$.

2. O latice pentru care este verificată relația

$$x \sqcup (y \sqcap z) = (x \sqcup y) \sqcap z$$

dacă $x \sqsubseteq z$, se numește latice modulară;

Să se arate că orice latice distributivă este modulară.

3. Fie $A = \{a_1, a_2, \dots, a_n\}$, o mulțime de numere reale pe care se definesc operațiile \sqcup și \sqcap astfel

$$a \sqcup b = \max(a, b) \text{ și } a \sqcap b = \min(a, b)$$

a) Să se arate că A este o latice distributivă

b) În ce condiții A este o algebră booleană?

4. Fie M mulțimea divizorilor lui 30.

a) Să se arate că laticea $\langle M, \text{cmmdc}, \text{cmmmc} \rangle$ este o algebra booleană

b) Câte elemente are mulțimea M ?

c) Care este complementul lui 6?

5. Să se arate ca în orice algebra booleană sunt adevărate următoarele relații.

a) $aa = a,$

b) $\overline{\overline{a}} = a,$ (principiul dublei negații)

c) $a0 = 0,$

d) $a + 1 = 1,$

e) $\overline{0} = 1,$

f) $\overline{1} = 0,$

g) $\overline{a + b} = \overline{a} \cdot \overline{b}$ (formulele lui De Morgan)

h) $\overline{\overline{a} \cdot \overline{b}} = \overline{\overline{a}} + \overline{\overline{b}}$

6. Să se arate că mulțimea poligoanelor convexe din plan pentru care se definesc operațiile \sqcap și \sqcup astfel:

$$P_1 \sqcap P_2 = P_1 \cap P_2 \text{ (intersecția celor două poligoane),}$$

$$P_1 \sqcup P_2 = \text{cel mai mic poligon convex care conține pe } P_1 \cup P_2,$$

este o latice. Care sunt primul și ultimul element al acestei latice?

7. Să se arate că într-o latice L cele două condiții de distributivitate sunt echivalente

8. Să se arate că într-o algebră booleană elementul nul a este prim element, iar elementul universal u este ultim element.

9. Să se efectueze următoarele calcule în algebra B_2 :

a) $(1 \cdot 0 + 1 \cdot 1) \cdot (1 + 0 \cdot 1),$

b) $((\overline{1 \cdot 0} + 1) + 0) \cdot 1,$

c) $(0 + \overline{1} + \overline{0})(1 + 0)$

1.2. Funcții booleene

1.2.1. Expresii booleene

Din aritmetica și algebra se știe că o expresie este un ansamblu de elemente (numere, litere) legate între ele prin simboluri care reprezintă operații matematice. Pentru a indica ordinea în care se efectuează operațiile, expresiile pot conține și paranteze.

În mod asemănător, se pot construi expresii cu elemente dintr-o algebră booleană, utilizând simbolurile celor trei operații \cdot , $+$ și $'$.

Definiție. Se numește **expresie booleană** orice expresie rezultată prin aplicarea de un număr finit de ori a operațiilor \cdot , $+$, $'$ unor elemente determinate sau nedeterminate ale unei algebre booleene.

Exemple

1. $(a \cdot b) + (\overline{c \cdot 1})$ este o expresie booleană care indică efectuarea operației „ \cdot ” între elementele a și b , a operației „ \cdot ” între elementele c și 1 , a operației de complementare a rezultatului ultimei operații și a operației „ $+$ ” între primul rezultat și ultimul.
2. Nu orice înșiruire de simboluri poate reprezenta o expresie. Astfel, $+ a \cdot + b$, nu este o expresie booleană, deoarece nu respectă definiția. Simbolurile operațiilor care apar în acest șir nu leagă elementele în mod corect (există doi operatori consecutivi).

Ordinea în care trebuie efectuate operațiile dintr-o expresie este determinată de parantezele care apar în cadrul expresiei. Scrierea unei expresii poate fi simplificată prin omiterea unor paranteze dacă se introduc reguli de prioritate a operațiilor. De obicei, operația de complementare are prioritatea cea mai mare, fiind urmată de operația „ \cdot ”, cea mai puțin prioritară fiind operația „ $+$ ”. La fel ca în algebra clasică, în expresiile booleene se poate omite simbolul operației „ \cdot ”.

Astfel, expresia $(a \cdot b) + (c \cdot 1)$ se poate scrie $ab + c1$.

Definiție. Fie o expresie booleană cu variabilele x_1, x_2, \dots, x_n . Se numește **valoare a expresiei booleene**, pentru șirul de valori v_1, v_2, \dots, v_n , valoarea obținută prin înlocuirea în expresie a variabilelor x_1, x_2, \dots, x_n cu valorile corespunzătoare v_1, v_2, \dots, v_n și efectuarea operațiilor indicate de simbolurile din expresie.

Exemplu. Valoarea expresiei $x_1 x_2 + \overline{x_1 x_2} + \overline{x_1} x_2$, pentru șirul de valori $0, 1, 0$ se obține astfel

$$0 \cdot 0 + 1 \cdot 0 + 0 \cdot 1 = 0 + 0 + 1 = 0 + 1 = 1.$$

Definiție. Două expresii booleene cu aceleași variabile se numesc **egale** (sau **echivalente**) dacă pentru aceleași șiruri de valori ale variabilelor iau valori egale.

Dacă numărul de elemente din algebra booleană este mic, se poate demonstra egalitatea a două expresii booleene prin verificarea egalității valorilor lor pentru toate șirurile de valori ale variabilelor.

Exemplu. În algebra booleană B_2 expresia $E_1(x, y) = \overline{x}y + y$ este egală cu expresia $E_2(x, y) = x + y$, deoarece:

$$E_1(0, 0) = 0\overline{0} + 0 = 0 \cdot 1 + 0 = 0 + 0 = E_2(0, 0);$$

$$E_1(0, 1) = 0\overline{1} + 1 = 0 \cdot 0 + 1 = 0 + 1 = E_2(0, 1);$$

$$E_1(1, 0) = 1\overline{0} + 0 = 1 \cdot 1 + 0 = 1 + 0 = E_2(1, 0);$$

$$E_1(1, 1) = 1\overline{1} + 1 = 1 \cdot 0 + 1 = 0 + 1 = 1 + 1 = E_2(1, 1).$$

Egalitatea a două expresii booleene poate fi demonstrată și prin calcul boolean folosind proprietățile celor trei operații booleene.

Astfel, pentru expresiile E_1 și E_2 considerate anterior se poate demonstra egalitatea și prin :

$$\begin{aligned}
 E_1(x, y) &= x\bar{y} + y = \\
 &= (x + y)(\bar{y} + y) \quad (\text{comutativitate, distributivitate}) \\
 &= (x + y)1 = \quad (\text{principiul terțului exclus}) \\
 &= x + y \quad (1 \text{ este element neutru pentru „+”}) \\
 &= E_2(x, y)
 \end{aligned}$$

Ca reguli de calcul boolean pot fi folosite atât relațiile din definiția algebrei booleene cât și alte relații stabilite pe baza acestora.

Exemple de calcul în algebra booleană.

1. $\overline{x + y + z}$
 $\quad \quad \quad \overline{(x + y) + z}$ asociativitatea adunării
 $\quad \quad \quad = \overline{(x + y)}z$ formula lui De Morgan
 $\quad \quad \quad = (\overline{x}\overline{y})z$ formula lui De Morgan
 $\quad \quad \quad = \overline{x}y\overline{z}$ asociativitatea înmulțirii.
2. $abc + \bar{a}bc + ab\bar{c} = ac(b + \bar{b}) + ab \cdot 0 = ac \cdot 1 + 0 = ac$

12.2. Funcții booleene

În general, o funcție este definită ca o lege care asociază fiecărui element al unei mulțimi (numită domeniu de definiție) un singur element dintr-o altă mulțime (numită domeniu de valori). În concordanță cu mulțimile care alcătuiesc domeniul de definiție și domeniul de valori, funcțiile pot fi, de exemplu, funcții întregi de variabilă întreagă, funcții raționale de variabilă întreagă, funcții reale de variabilă reală. În cele ce urmează, se vor considera funcții la care atât domeniul de definiție cât și domeniul de valori sînt algebre booleene. Deoarece în practică algebra booleană cea mai răspîndită este algebra B_2 , funcțiile booleene care vor fi studiate vor fi definite astfel :

Definiție. Se numește **funcție booleană** (sau **funcție binară**) de n variabile, o funcție definită pe produsul cartezian $\underbrace{B_2 \times B_2 \times \dots \times B_2}_{\text{de } n \text{ ori}}$ cu valori în mulțimea B_2 .

Deoarece domeniul de definiție este un produs cartezian, un element al domeniului de definiție este un sistem ordonat alcătuit din n elemente din algebra B_2 , adică din n elemente 0 sau 1. Pentru fiecare combinație de n elemente 0 sau 1, funcția îi asociază un element din algebra B_2 (adică 0 sau 1), numit **valoarea funcției** booleene pentru sistemul de n elemente considerat. O variabilă care ia valori în algebra B_2 se numește **variabilă booleană**. Putem astfel considera că funcțiile booleene sînt funcții de una sau mai multe variabile booleene.

Exemplu. Putem defini o funcție $f: B_2 \times B_2 \rightarrow B_2$, astfel :

$$f(0, 0) = 0, \quad f(0, 1) = 0, \quad f(1, 0) = 0, \quad f(1, 1) = 1$$

deoarece $B_2 \times B_2 = \{(0, 0), (0, 1), (1, 0), (1, 1)\}$. Această funcție poate fi reprezentată prin diagrama din figura I.4.

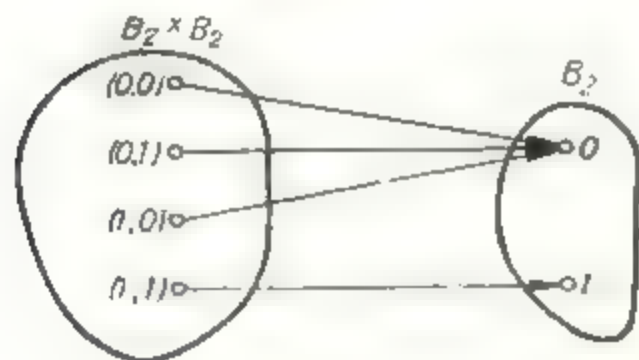


Fig. I.4

1.2.3. Reprezentarea funcțiilor booleene

Cea mai simplă formă de reprezentare a unei funcții booleene este tabelul său de valori. Domeniul de definiție al unei funcții booleene, B_2^n , are 2^n elemente astfel că tabelul va conține cele 2^n valori corespunzătoare ale funcției.

Exemple

1. Tabelul de valori pentru funcția din figura 1.4 este

x_1x_2	00	01	10	11
f	0	0	0	1

Tabelul 1.2

O altă formă a aceluiași tabel este următoarea :

$x_1 \backslash x_2$	0	1
0	0	0
1	0	1

Tabelul 1.3

2. Pentru funcția din figura 1.5 tabelul de valori poate fi :

$x_1x_2x_3$	000	001	010	011	100	101	110	111
g	1	0	0	0	1	0	1	1

Tabelul 1.4

sau

$x_1 \backslash x_2x_3$	00	01	10	11
0	1	0	0	0
1	1	0	1	1

Tabelul 1.5

3. Tabelul de valori poate fi alcătuit prin utilizarea unui cod ciclic pentru șirul de valori ale variabilelor :

x_1x_2	00	01	11	10
f	0	0	1	0

Tabelul 1.6

Se poate arăta că pentru orice funcție booleană există o expresie booleană care o generează.

Exemple

1. Funcția $f: B_2^2 \rightarrow B_2$ din figura 1.4 poate fi generată de expresia $x_1 x_2$. Aceasta se poate scrie:

$$f(x_1, x_2) = x_1 x_2.$$

2. Funcția $g: B_2^3 \rightarrow B_2$ din figura 1.5 poate fi generată de expresia $x_1 x_2 + \bar{x}_2 \bar{x}_3$. Deci

$$g(x_1, x_2, x_3) = x_1 x_2 + \bar{x}_2 \bar{x}_3.$$

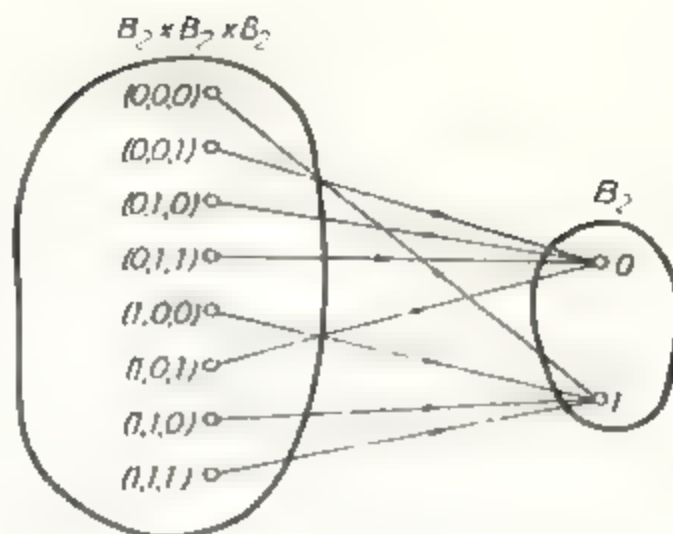


FIG. 1.5

Să considerăm expresia $(x_1 + \bar{x}_2)(x_2 + \bar{x}_3)$. Aceasta expresie generează o funcție al cărei tabel de valori este:

$x_1 x_2 x_3$	\bar{x}_2	$x_1 + \bar{x}_2$	\bar{x}_3	$x_2 + \bar{x}_3$	$(x_1 + \bar{x}_2)(x_2 + \bar{x}_3)$
000	1	1	1	1	1
001	1	1	0	0	0
010	0	0	1	1	0
011	0	0	0	1	0
100	1	1	1	1	1
101	1	1	0	0	0
110	0	1	1	1	1
111	0	1	0	1	1

Tabelul 1.9

Se observa că această funcție coincide cu funcția generată de expresia $x_1 x_2 + \bar{x}_2 \bar{x}_3$. Rezultă deci, că aceeași funcție poate fi generată de mai multe expresii booleane.

Cele 16 funcții de două variabile, prezentate în tabelul 1.8, pot fi generate de următoarele expresii: $f_0 = 0$; $f_1 = x_1 x_2$; $f_2 = x_1 \bar{x}_2$; $f_3 = x_1$; $f_4 = \bar{x}_1 x_2$; $f_5 = \bar{x}_1$; $f_6 = \bar{x}_1 x_2 + x_1 \bar{x}_2$; $f_7 = x_1 + x_2$; $f_8 = \bar{x}_1 \bar{x}_2$; $f_9 = \bar{x}_1 \bar{x}_2 + x_1 x_2$; $f_{10} = \bar{x}_2$; $f_{11} = x_1 + \bar{x}_2$; $f_{12} = \bar{x}_1$; $f_{13} = \bar{x}_1 + x_2$; $f_{14} = x_1 + x_2$; $f_{15} = 1$.

Funcțiile f_0 , f_2 , f_4 , f_{10} , f_{12} și f_{14} poartă denumirea de funcții booleane degenerate de două variabile, deoarece valoarea lor nu depinde de ambele variabile. Celelalte funcții poartă denumiri variate, care sînt provenite din teoria mulțimilor sau din calculul propozițional.

Astfel, se întâlnesc următoarele denumiri:

- f_1 — intersecție, conjuncție, și, produs;
- f_2, f_4 — intersecție indirectă;
- f_6 — sumă disjunctivă, sau exclusiv, diferență simetrică;
- f_7 — sumă logică, reuniune, disjuncție, sau inclusiv;
- f_8 — funcția nici, funcția lui Peirce;
- f_9 — echivalență;
- f_{11}, f_{13} — implicație;
- f_{14} — excludiune, funcția lui Sheffer.

Studierea funcțiilor booleene de două variabile are o importanță deosebită, deoarece se demonstrează că funcțiile booleene de trei sau mai multe variabile pot fi descompuse în funcții booleene de două variabile. De exemplu, funcția $g(x_1, x_2, x_3) = x_1x_2 + \overline{x_2}x_3$ poate fi descompusă astfel:

$$g(x_1, x_2, x_3) = f_7(f_1(x_1, x_2), f_2(x_2, x_3)).$$

Așadar, orice funcție booleană poate fi reprezentată cu ajutorul funcțiilor booleene de două variabile.

Dacă se ține seama de faptul că orice expresie booleană este formată utilizând cele trei operații (\cdot , $+$ și \neg) și că aceste operații corespund funcțiilor f_1 , f_2 și f_{12} , rezultă că sînt suficiente aceste trei funcții booleene de două variabile pentru a reprezenta orice funcție booleană. În aceste condiții, funcția g scrisă mai înainte poate fi descompusă astfel:

$$g(x_1, x_2, x_3) = f_7(f_1(x_1, x_2), f_1(f_{12}(x_2, 0), f_{12}(x_3, 0)))$$

Utilizînd formulele lui De Morgan se poate scrie că:

$$f_7(x_1, x_2) = f_{12}(f_{12}(f_{12}(x_1, 0), f_{12}(x_2, 0)), 0) \text{ și}$$

$$f_1(x_1, x_2) = f_{12}(f_2(f_{12}(x_1, 0), f_{12}(x_2, 0)), 0),$$

ceea ce permite să se afirme că orice funcție booleană poate fi reprezentată cu ajutorul a numai două funcții booleene de două variabile: fie f_1 și f_{12} , fie f_2 și f_{12} . Aceeași funcție g considerată anterior, se poate scrie de exemplu:

$$g(x_1, x_2, x_3) = f_{12}(f_1(f_1(f_1(x_1, 0), 0), f_1(f_1(x_2, 0), f_1(x_3, 0)))), 0)$$

O proprietate remarcabilă a au însă funcțiile f_2 și f_{12} . Relațiile

$$f_1(x_1, x_2) = f_2(f_2(x_1, 0), f_2(x_2, 0));$$

$$f_2(x_1, x_2) = f_2(f_2(x_1, x_2), 0);$$

$$f_{12}(x_1) = f_2(x_1, 0)$$

permît ca prin înlocuirea funcțiilor f_1 , f_2 și f_{12} cu expresiile indicate să se poată găsi pentru orice funcție booleană o reprezentare care să utilizeze numai funcția f_2 (funcția lui Peirce). Astfel, funcția g poate fi reprezentată sub forma:

$$g(x_1, x_2, x_3) = f_2(f_2(f_2(f_2(x_1, 0), f_2(x_2, 0)), f_2(x_2, x_3)), 0),$$

deoarece $g(x_1, x_2, x_3) = f_7(f_1(x_1, x_2), f_2(x_2, x_3))$

Funcția lui Peirce fiind o funcție binară este notată uneori ca o operație binară, utilizîndu-se simbolul \downarrow . Cu aceasta notație putem reprezenta funcția g astfel:

$$g(x_1, x_2, x_3) = (((x_1 \downarrow 0) \downarrow (x_2 \downarrow 0)) \downarrow (x_2 \downarrow x_3)) \downarrow 0.$$

În mod analog se poate reprezenta orice funcție booleană utilizînd numai funcția lui Sheffer (vezi ex. 8).

Funcțiile lui Peirce și Sheffer nu au numai o importanță teoretică ci și una practică, deoarece circuitele electronice care realizează funcțiile lui Peirce și Sheffer sînt mai simple decît cele care realizează operațiile booleene $+$ și \cdot .

1.2.4. Exerciții

1. Să se restrîngă următoarele expresii:

a) $(x + xy)(x + yz)$;

b) $a(\bar{a} + b) + b(b + c) + \bar{b}$;

c) $(a + b)(a + \bar{b})(\bar{a} + \bar{b})(\bar{a} + b)$;

d) $xyz + x\bar{y}z + xy\bar{z} + x\bar{y}\bar{z}$.

2. Să se verifice următoarele identități:

- a) $a + \overline{ab} = a + b$;
- b) $(a + \overline{c})(b + c) = ac + b\overline{c}$;
- c) $ab + bc + ca = (a + b)(b + c)(c + a)$.

3. Fie expresiile:

$$\begin{aligned} E_1 &= (a + ab)(a + b); \\ E_2 &= \overline{ab} + \overline{(a + b + c + d)}; \\ F_3 &= \overline{(a + b)(c + d)}; \\ E_4 &= a + b\overline{c} + \overline{abc}(ad + b). \end{aligned}$$

Să se arate că:

- a) $E_1 = a$; b) $E_2 = \overline{E_3}$; c) $E_3 = E_2 + \overline{cd}$; d) $E_4 = a + b$.

4. Să se calculeze valorile expresiilor:

- a) $(ab + c)(\overline{b} + c)$ pentru $a = 0, b = 1$ și $c = 1$;
- b) $ab(\overline{a} + c) + \overline{ab}$ pentru $a = 1, b = 1$ și $c = 0$;
- c) $\overline{ab + c + \overline{ac}}$ pentru $a = 0, b = 0$ și $c = 0$.

5. Să se verifice următoarele relații din algebra B , înlocuind nedeterminatele cu toate valorile posibile;

- a) $\overline{ab} + a\overline{b} + ab = a + b$;
- b) $(a + b)(a + \overline{b}) = a$;
- c) $\overline{ac} + b + ac = a + b$;
- d) $ab + bc + \overline{ac} = ab + \overline{ac}$.

6. Să se afle care dintre următoarele expresii booleene sînt egale.

- a) $E_1 = ab + bc + ac$; $E_2 = \overline{abc} + a\overline{bc} + ab$;
- b) $E_1 = \overline{abc} + \overline{abd} + abc + abd$; $E_2 = \overline{bcd} + \overline{acd} + bcd + acd$;
- c) $E_1 = \overline{xz} + \overline{xy} + x\overline{y}$; $E_2 = (\overline{x} + \overline{y})(x + y + \overline{z})$.

7. Să se arate că:

- a) $x\overline{z} + \overline{xz} = y$ dacă $x\overline{y} + \overline{xy} = z$;
- b) $ad + \overline{b}(\overline{c} + \overline{d}) = ad + ac + \overline{ac} + \overline{bcd}$ dacă $a\overline{d} + bc = 0$.

8. Să se alcătuiască tabelul de valori al următoarelor funcții:

- a) $f(x_1, x_2) = x_1x_2 + x_2$;
- b) $g(x_1, x_2) = (x_1 + x_2)(\overline{x_1} + x_2x_2)$;
- c) $f(x, y, z) = xz + y\overline{z} + \overline{xy}$;
- d) $g(x, y, z) = \overline{x}(y + \overline{z}) + \overline{y}(x + z)$.

9. Se consideră funcțiile:

$$\begin{aligned} g_1(a, b, c) &= ab + bc + c\overline{a}; \\ g_2(a, b, c) &= (a + c)(b + \overline{a}); \end{aligned}$$

$$g_1(a, b, c) = ab + c\bar{a};$$

$$g_2(a, b, c) = (a + b)(b + c)(c + \bar{a});$$

$$g_3(a, b, c) = ac + b\bar{a};$$

$$g_4(a, b, c) = (a + b)(c + \bar{a}).$$

Să se arate că :

a) funcțiile g_1 , g_2 și g_3 sînt egale,

b) funcțiile g_1 , g_2 , g_3 sînt egale,

1.3. Forme canonice ale funcțiilor booleene

1.3.1. Forme normale ale funcțiilor booleene

În paragraful anterior s-a arătat că o aceeași funcție booleană poate fi reprezentată prin mai multe expresii booleene. Este de dorit să existe totuși o formă standardizată de reprezentare prin expresii a unei funcții booleene. Această formă de reprezentare trebuie să fie unică, pentru a permite determinarea rapidă a egalității a două funcții prin compararea expresiilor asociate.

Definiție. Se numește **produs elementar** un produs de variabile booleene sau complemente ale acestora, fără ca aceeași variabilă să apară de mai multe ori.

Exemplu $\bar{a}bc$, $\bar{a}bc$, xyz , $\bar{x}_1\bar{x}_2$ sînt produse elementare iar $a + bc$, $\bar{x}a$ nu sînt produse elementare

Definiție. Se numește **formă normală disjunctivă** a unei expresii booleene o sumă de produse elementare egală cu expresia dată.

Exemplu. $ab + bc + ac$, $\bar{x}\bar{y}\bar{z} + xyz$, $x_1\bar{x}_2$ sînt expresii sub formă normală disjunctivă.

Vom spune că o funcție booleană este scrisă sub forma normală disjunctivă dacă este reprezentată printr-o expresie sub forma normală disjunctivă

Forma normală disjunctivă a unei funcții nu este unică, după cum se poate observa din exemplul următor

$$f(x, y, z) = xy + yz + \bar{x}z = xy + \bar{x}z = xy + xyz + \bar{x}\bar{z}.$$

Putem astfel obține oricâte forme normale disjunctive utilizînd idempotența și absorbția

Observație. Condiția ca într-un produs elementar să nu se repete variabilele are o justificare simplă. Dacă aceeași variabilă apare de mai multe ori, atunci aceasta poate fi redusă la o singură apariție prin idempotență. Dacă aceeași variabilă apare împreună cu complementul său atunci produsul respectiv este egal cu 0, astfel că poate fi eliminat din sumă.

În mod analog se definesc suma elementară și forma normală conjunctivă a unei funcții.

Definiție. Se numește **sumă elementară** o sumă de variabile booleene sau complemente ale acestora, fără ca aceeași variabilă să apară de mai multe ori

De exemplu, $\bar{a} + \bar{b} + c$, $\bar{a} + b + c$, $x + y + z$, $\bar{x}_1 + \bar{x}_2$ sînt sume elementare iar $a + bc$, $\bar{x} + y$ nu sînt.

Definiție. Se numește **formă normală conjunctivă** a unei expresii booleene un produs de sume elementare egal cu expresia dată.

De exemplu $(a + b)(b + c)(a + c)$, $(\bar{x} + \bar{y} + \bar{z})(x + y + z)$, $x_1 + \bar{x}_2$ sînt expresii sub forma normală conjunctivă.

Nici forma normală conjunctivă a unei expresii nu este unică, după cum rezultă din exemplul următor :

$$(x + y)(y + z)(\bar{x} + z) = (x + y)(x + z) = (x + y)(x + y + z)(\bar{x} + z)$$

Orice expresie booleană (deci și orice funcție) poate fi adusă la o formă normală disjunctivă și la o formă normală conjunctivă.

Pentru a aduce o expresie booleană la o formă normală disjunctivă se folosește următorul procedeu :

— dacă în cadrul expresiei operația de complementare este aplicată unor expresii, se aplică formulele lui De Morgan, pînă cînd în cadrul expresiei date nu mai apar decît complementele variabilelor ;

— se distribuie operația „ \cdot ” în raport cu „ $+$ ” ori de cîte ori este cazul ;

— se elimină produsele care se anulează sau se repeta și variabilele care apar de două ori în același produs.

Exemplu. Expresia $F = xy \cdot (xz + \overline{x + z}) + x\bar{y}z$ poate fi transformată astfel :

$$E = (\bar{x} + \bar{y})(x\bar{z} + \bar{x}z) + x\bar{y}z \text{ (s-au aplicat formulele lui De Morgan)}$$

$$= x\bar{x}z + x\bar{x}z + x\bar{y}z + x\bar{y}z + x\bar{y}z \text{ (s-a aplicat distributivitatea operației } \cdot \text{ față de } +)$$

$$= \bar{x}z + x\bar{y}z + x\bar{y}z \text{ (s-au eliminat primul și ultimul produs).}$$

S-a obținut pentru E o formă normală disjunctivă.

Dacă utilizăm și proprietatea de absorbție, putem elimina și ultimul produs, obținînd pentru expresia dată o altă formă normală

$$E = \bar{x}z + x\bar{y}z.$$

Pentru a aduce o expresie booleană la o formă normală conjunctivă se folosește un procedeu similar, în care, după aplicarea formulelor lui De Morgan, se distribuie operația „ $+$ ” în raport cu „ \cdot ” și se elimină sumele care au valoarea constantă 1 sau se repetă, precum și variabilele care apar de două ori în aceeași sumă.

Exemplu. Aceeași expresie E poate fi adusă la o formă normală conjunctivă astfel :

$$E = \bar{x}y(x\bar{z} + \overline{x + z}) + x\bar{y}z = (\bar{x} + \bar{y})(x\bar{z} + \bar{x}z) + x\bar{y}z =$$

$$= (\bar{x} + \bar{y})(x + z)(\bar{x} + \bar{z}) + x\bar{y}z =$$

$$= (\bar{x} + \bar{y} + x)(\bar{x} + \bar{y} + \bar{y})(\bar{x} + \bar{y} + \bar{z})(x + z + x)(x + z + \bar{y}) =$$

$$= (x + z + \bar{z})(\bar{x} + \bar{z} + x)(\bar{x} + \bar{z} + \bar{y})(\bar{x} + \bar{z} + \bar{z}) =$$

$$= (\bar{x} + \bar{y})(\bar{x} + \bar{y} + \bar{z})(x + z)(x + y + z)(\bar{x} + \bar{z}).$$

Expresia se poate simplifica prin absorbție astfel că

$$E = (\bar{x} + \bar{y})(x + z)(\bar{x} + \bar{z}).$$

1.3.2. Forme canonice ale funcțiilor booleene

Definiție. Se numește *mintermen* în raport cu variabilele booleene x_1, x_2, \dots, x_n un produs elementar în care apar, fie simple, fie complementate, toate variabilele x_1, x_2, \dots, x_n .

Textele prevăzute cu o bară laterală sînt destinate numai profilului de matematică.

Un mintermen are deci n factori. Acești factori se scriu în ordinea naturală a variabilelor care îi alcătuiesc. De exemplu,

$$x_1 x_2 \bar{x}_3, \bar{a} \bar{b} \bar{c}, \bar{x} \bar{y} z, \bar{a} b, x_1 x_2 x_3 x_4 x_5.$$

Fiecărui mintermen i se poate asocia un număr binar astfel:

Variabilele complementate se înlocuiesc cu cifra 0, iar variabilele necomplementate cu cifra 1. Șirul de cifre obținut reprezintă un număr binar (în baza 2), care este asociat mintermenului respectiv. De exemplu, mintermenilor scriși anterior li se asociază numerele:

$(110)_2, (100)_2, (000)_2, (11)_2, (11111)_2$, care reprezintă în baza zece numerele 6, 4, 0, 3, 31.

Considerind n variabile booleene, numărul de mintermeni diferiți care se pot construi cu aceste variabile este egal cu numărul de numere binare cu n cifre. Acest număr este 2^n .

Mintermenii joacă un rol important în cadrul formelor de reprezentare a funcțiilor booleene, datorită următoarei proprietăți.

Un mintermen are valoarea 1 pentru un singur șir de valori ale variabilelor booleene care îl alcătuiesc.

Într-adevăr, deoarece un produs boolean are valoarea 1 dacă și numai dacă toți factorii săi au valoarea 1, iar factorii unui mintermen sînt variabile booleene simple sau complementate, alegînd valoarea 0 pentru variabilele care apar complementate în cadrul mintermenului și valoarea 1 pentru celelalte variabile, se obține combinația de valori căutată.

Se definește în mod analog noțiunea de maxtermen

Definiție. Se numește **maxtermen** în raport cu variabilele booleene x_1, x_2, \dots, x_n o sumă elementară în care apar, fie simple, fie complementate, toate variabilele x_1, \dots, x_n .

Un maxtermen are deci n termeni, care se scriu în ordinea naturală a variabilelor care îi alcătuiesc. De exemplu

$$x_1 + \bar{x}_2 + x_3, \bar{x} + \bar{y} + z, \bar{a} + \bar{b} + c, x + y, x_1 + x_2 + x_3 + x_4 + x_5$$

Pentru un maxtermen se pot face aceleași observații ca pentru mintermeni. Asocierea unui număr binar pentru fiecare maxtermen se face înlocuind variabilele complementate cu cifra 1, iar variabilele necomplementate cu cifra 0. Numărul de maxtermeni care se pot forma cu n variabile booleene este, de asemenea, 2^n .

Maxtermenii au o proprietate similară cu aceea a mintermenilor, și anume:

Un maxtermen are valoarea 0 pentru o singură combinație de valori ale variabilelor booleene, care îl alcătuiesc (0 pentru variabilele simple și 1 pentru variabilele complementate).

Mintermenii permit alcătuirea unor forme normale particulare, deosebit de importante.

Definiție. Se numește **formă canonică disjunctivă** a unei expresii booleene cu n variabile o formă normală disjunctivă echivalentă cu acea expresie, alcătuită numai din mintermeni cu n variabile.

Forma canonică disjunctivă a unei expresii se mai numește și **formă normală disjunctivă perfectă**.

Exemple. $\bar{a} \bar{b} c + \bar{a} b c + a b c, x y z + \bar{x} \bar{y} \bar{z} + x \bar{y} z, x_1 x_2 \bar{x}_3 x_4 + \bar{x}_1 \bar{x}_2 x_3 x_4 + x_1 x_2 x_3 x_4$ sînt forme canonice disjunctive.

În aceste exemple, primele 2 expresii au câte 3 variabile, iar ultima expresie are 4 variabile.

Propoziție. Forma canonică disjunctivă a unei funcții booleene este unică, dacă facem abstracție de ordinea mintermenilor.

Demonstrație. Să presupunem că pentru o funcție booleană există două forme canonice disjunctive distincte.

Deci există un mintermen care face parte dintr-una din formele canonice (să spunem din prima formă) și nu face parte din cealaltă. Să considerăm combinația de valori ale variabilelor pentru care mintermenul respectiv are valoarea 1 și să analizăm valorile celor două expresii corespunzătoare acestei combinații. Deoarece pentru această combinație de valori ale variabilelor mintermenul din prima expresie are valoarea 1, indiferent de valorile celorlalți mintermeni, prima expresie are valoarea 1. Pentru a doua expresie se obține însă valoarea 0, deoarece toți mintermenii acestei expresii nu pot avea decât valoarea 0 (conform proprietății mintermenilor prezentată anterior). Rezultă, deci, că aceste două expresii nu sunt echivalente, adică presupunerea făcută este falsă.

Cele două forme canonice conțin aceeași mintermeni, ceea ce încheie demonstrația.

Forma canonică disjunctivă poate fi deci considerată ca o reprezentare standard a funcțiilor booleene.

Totuși, există o singură funcție booleană care nu poate fi reprezentată sub forma canonică disjunctivă și anume funcția constantă 0. Aceasta este o consecință a faptului că fiecărui mintermen dintr-o formă canonică disjunctivă îi corespunde o combinație de valori ale variabilelor pentru care funcția respectivă ia valoarea 1, astfel că o funcție reprezentată sub forma canonică disjunctivă ia cel puțin o dată valoarea 1.

Forma canonică disjunctivă a unei funcții booleene poate fi obținută atât pe baza tabelului de valori a funcției, cât și pe baza unei forme normale disjunctive a acesteia.

Tabela de valori a unei funcții booleene permite obținerea formei canonice disjunctive a funcției astfel:

Pentru fiecare combinație de valori ale variabilelor booleene pentru care funcția are valoarea 1 se considera mintermenul corespunzător (adică acel mintermen care ia valoarea 1 numai pentru această combinație). Suma tuturor acestor mintermeni constituie forma canonică disjunctivă a funcției. Reamintim că mintermenul corespunzător unei combinații de valori ale variabilelor este produsul variabilelor care iau valoarea 1 și a complementelor variabilelor care iau valoarea 0.

Fie, de exemplu, funcția definită de următoarea tabelă de valori

<i>a</i>	0	0	0	0	1	1	1	1
<i>b</i>	0	0	1	1	0	0	1	1
<i>c</i>	0	1	0	1	0	1	0	1
<i>f</i>	1	0	0	0	1	0	1	1

Tabelul 1.10

Funcția ia de 4 ori valoarea 1, pentru următoarele combinații de valori ale variabilelor *a*, *b*, *c*: (0, 0, 0), (1, 0, 0), (1, 1, 0), (1, 1, 1)

Acesteor combinații de valori le corespund următorii mintermeni: $\bar{a}\bar{b}\bar{c}$, $\bar{a}b\bar{c}$, $a\bar{b}\bar{c}$, abc . Forma canonică disjunctivă a acestei funcții este deci

$$f(a, b, c) = \bar{a}\bar{b}\bar{c} + \bar{a}b\bar{c} + a\bar{b}\bar{c} + abc.$$

O formă normală disjunctivă a unei funcții poate fi transformată într-o formă canonică disjunctivă astfel:

Fiecare termen al formei normale disjunctive care nu este mintermen, adică fiecare termen care nu conține toate variabilele, se înmulțește cu expresii de forma $(x + \bar{x})$, pentru fiecare variabilă x absentă din acel termen. Ținând cont de distributivitatea înmulțirii booleene față de adunarea booleană se desfac parantezele și se elimină termenii dubli (ținând seama de idempotența adunării booleene). Se poate observa ușor că expresia obținută nu conține decât mintermeni și este echivalentă cu expresia inițială. Rezultă deci că s-a obținut forma canonică a funcției respective.

Exemplu. Fie forma normală disjunctivă:

$$f(a, b, c) = \bar{b}\bar{c} + a\bar{c} + ab.$$

Aceasta se transformă astfel:

$$\begin{aligned}\bar{b}\bar{c} + a\bar{c} + ab &= \bar{b}\bar{c}(a + \bar{a}) + a\bar{c}(b + \bar{b}) + ab(c + \bar{c}) = \\ &= \bar{a}\bar{b}\bar{c} + \bar{a}b\bar{c} + a\bar{b}\bar{c} + a\bar{b}c + abc + ab\bar{c} = \\ &= \bar{a}\bar{b}\bar{c} + \bar{a}b\bar{c} + a\bar{b}\bar{c} + abc.\end{aligned}$$

În mod analog, utilizând noțiunea de maxtermen, se definește forma canonică conjunctivă a unei expresii booleene. Forma canonică conjunctivă permite de asemenea o reprezentare standard a funcțiilor booleene deoarece și această formă are proprietatea de unicitate. Singura funcție care nu poate fi reprezentată sub formă canonică conjunctivă este funcția constantă 1.

De asemenea, forma canonică conjunctivă se poate obține pe baza fie a tabelului de valori a funcției, fie a unei forme normale conjunctive a acesteia.

1.3.3. Exerciții

- Să se găsească o formă normală disjunctivă a următoarelor expresii.
 - $(x + xy)(x + y)$;
 - $(\overline{x_1 x_2})(\overline{x_1 + x_2})(\overline{x_3 + x_4})$;
 - $\overline{(a + b)(c + d)}$;
 - $(a + b)\bar{c} + \bar{a}bc(ad + b)$.
- Să se găsească o formă normală conjunctivă a expresiilor de la exercițiul 1.
- Să se indice care dintre următoarele forme normale sînt și canonice.
 - $f(x, y, z) = x\bar{y} + y\bar{z}$;
 - $f(a, b, c) = abc + \bar{a}\bar{b}\bar{c} + \bar{a}bc + \bar{a}b\bar{c}$;
 - $f(x_1, x_2, x_3, x_4) = x_1 x_2 x_3 x_4 + \bar{x}_1 \bar{x}_2 x_3 \bar{x}_4 + x_1 \bar{x}_2 x_3 x_4$;
 - $g(x, y) = x + \bar{y}$;

e) $g(x, y) = xy$;

f) $f(a, b) = (a + b)(\bar{a} + \bar{b})$;

g) $g(x, y, z) = (\bar{x} + \bar{y} + \bar{z})(y + z)(x + y + z)$;

h) $f(x_1, x_2, x_3, x_4) = (x_1 + x_2 + x_3 + x_4\bar{x}_4)(\bar{x}_1 + x_2 + \bar{x}_3 + x_4)$.

4. Să se afle forma canonică disjunctivă a următoarelor expresii:

a) $ab + \bar{b}c$,

b) $\bar{x}\bar{y}\bar{z} + yz$;

c) $\bar{x}_1x_2 + \bar{x}_1x_3$.

5. Să se găsească forma canonică conjunctivă a expresiilor de la exercițiul 4

6. Fie funcția $f(a, b, c, d)$ definită de tabelul 1.11.

a) Să se scrie forma canonică disjunctivă a acestei funcții

b) Să se scrie forma canonică conjunctivă a funcției f .

c) Să se verifice prin calcul boolean că expresiile obținute la punctele a) și b) sînt echivalente.

a	b	c	d	$f(a, b, c, d)$
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

Tabelul 1. 11

1.4. Simplificarea funcțiilor booleene

1.4.1 Simplificarea prin calcul boolean

O problemă importantă legată de reprezentarea funcțiilor booleene o constituie găsirea unei forme cât mai simple pentru expresiile acestora.

Simplificarea prin calcul boolean este asemănătoare cu simplificarea expresiilor algebrice obișnuite. Trebuie însă să se țină cont de regulile specifice ale calculului boolean.

În acest sens sînt deosebit de utile regulile de idempotență ($a + a = a$ și $aa = a$), de absorbție ($a + ab = a$, $a(a + b) = a$) și de combinare ($ab + a\bar{b} = a$, $(a + b)(a + \bar{b}) = a$).

Aceste reguli pot fi folosite atît ca reguli de simplificare, cît și ca artificii de calcul pentru introducerea unor noi termeni necesari pentru combinațiile ulterioare.

Exemple

$$1. abcd + ab\bar{c} + abc + abd + \bar{a}bcd + \bar{a}bd =$$

$$= ab\bar{c} + abc + abd + \bar{a}bd = (\text{prin absorbție})$$

$$= ab + bd (\text{prin combinare}).$$

$$2. ab + \bar{a}bc + a\bar{b}c =$$

$$= ab + abc + \bar{a}bc + a\bar{b}c = (\text{prin absorbție}) (\text{artificiu de calcul})$$

$$= ab + abc + \bar{a}bc + abc + a\bar{b}c = (\text{prin idempotență}) (\text{artificiu de calcul})$$

$$= ab + (a + \bar{a})bc + ac(b + \bar{b}) =$$

$$= ab + bc + ac (\text{prin combinare}).$$

Totuși calculul boolean direct nu asigură obținerea celei mai simple expresii. Afiarea celei mai simple expresii trebuie să țina seama de toate combinațiile posibile, efectuînd o căutare metodică. Trebuie stabilit, însa, ce se înțelege prin expresie mai simplă.

O expresie va fi considerată mai simplă decît alta dacă numărul de apariții ale variabilelor din prima expresie este mai mic decît numărul corespunzător din cealaltă.

Astfel expresia $ab + \bar{a}c$ (în care variabilele apar de patru ori) va fi considerată mai simplă decît $a + b + \bar{a}bc$ în care variabilele apar de cinci ori.

De obicei se caută cea mai simplă expresie în forma normală disjunctivă, echivalentă cu expresia dată.

1.4.2. Simplificarea prin diagrame

Utilizarea diagramelor pentru simplificarea funcțiilor booleene este o metodă utilizată de obicei pentru funcțiile reprezentate prin tabele.

Diagramele Euler-Venn sînt cunoscute din teoria mulțimilor. S-a arătat însă că $\langle \mathcal{P}(M), \cap, \cup \rangle$ este o algebră booleană, astfel ca aceste diagrame pot fi utilizate și la reprezentarea geometrică a relațiilor din algebra booleană B_2 . Într-o diagramă Euler-Venn fiecare variabilă a expresiei booleene se reprezintă printr-o mulțime, de obicei un cerc. Interiorul mulțimii corespunde variabilei respective, iar exteriorul mulțimii (deci complementara mulțimii) corespunde complementului variabilei. Un termen al unei expresii booleene corespunde intersecției mulțimilor corespunzătoare variabilelor care îl compun. În cazul unei variabile care apare sub formă de complement se considera desigur intersecția cu exteriorul mulțimii corespunzătoare.

Întreaga expresie va corespunde astfel reuniunii mulțimilor corespunzătoare fiecărui termen.

În figurile 1.6, 1.7 și 1.8 sînt prezentate cîteva exemple de diagrame Euler-Venn, corespunzînd termenilor a , ab și, respectiv, abc . Se poate observa cã unui mintermen îi corespunde o zonã care nu mai este divizată în alte zone. În figura 1.9 au fost înscrîși mintermenii corespunzãtori fiecărei zone elementare. Considerînd expresia

$$abc + \bar{a}bc + ab\bar{c} + abc,$$

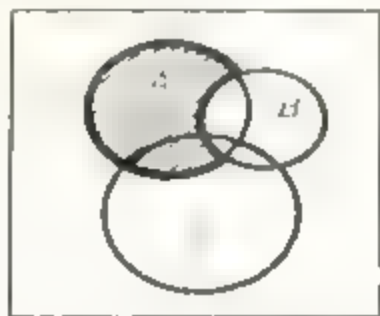


Fig. 1.6

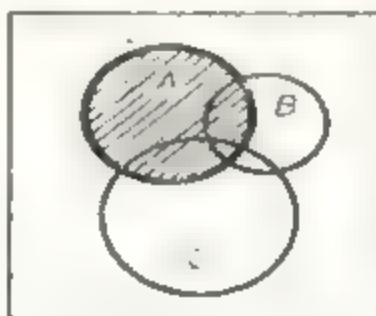


Fig. 1.7

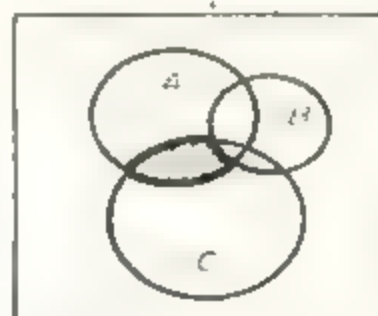


Fig. 1.8

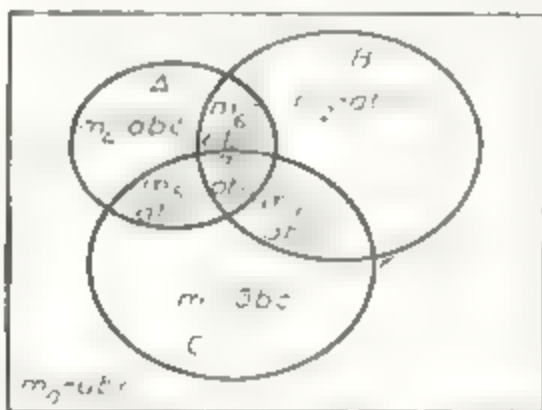


Fig. 1.9

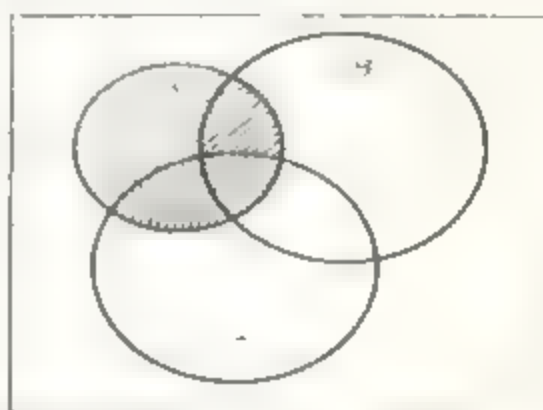


Fig. 1.10

se observã cã cei patru mintermeni corespund celor patru zone m_1 , m_2 , m_3 și m_4 . Reuniunea acestor patru zone (fig. 1.10) formeazã mulțimea corespunzãtoare variabilei a , deci

$$abc + \bar{a}bc + ab\bar{c} + abc = a.$$

Utilizarea diagramelor Euler-Venn devine dificilã dacã expresiile conțin mai mult de trei variabile.

Diagramele Veitch-Karnaugh reprezintã rearanjarea diagramelor Euler-Venn sub formã de tablou. În cazul diagramelor Veitch-Karnaugh mulțimile corespunzãtoare variabilelor booleene sînt reprezentate de dreptunghiuri. În figura 1.11 este prezentatã diagrama Veitch-Karnaugh pentru patru variabile. Se observã cã fiecare cãsuțã a diagramei corespunde unui mintermen. În figura 1.12 sînt prezentate diagramele Veitch-Karnaugh corespunzãtoare termenilor a , \bar{a} , bc , $\bar{b}c$, bcd , $\bar{a}bd$. Se observã cã mulțimile corespunzãtoare unor termenii sînt reprezentate prin unul sau mai multe pãtrate.

O proprietate importantã a diagramelor Veitch-Karnaugh este cã mintermenii care nu difera decît printr-un factor sînt reprezentați pe diagrama

Fig 111

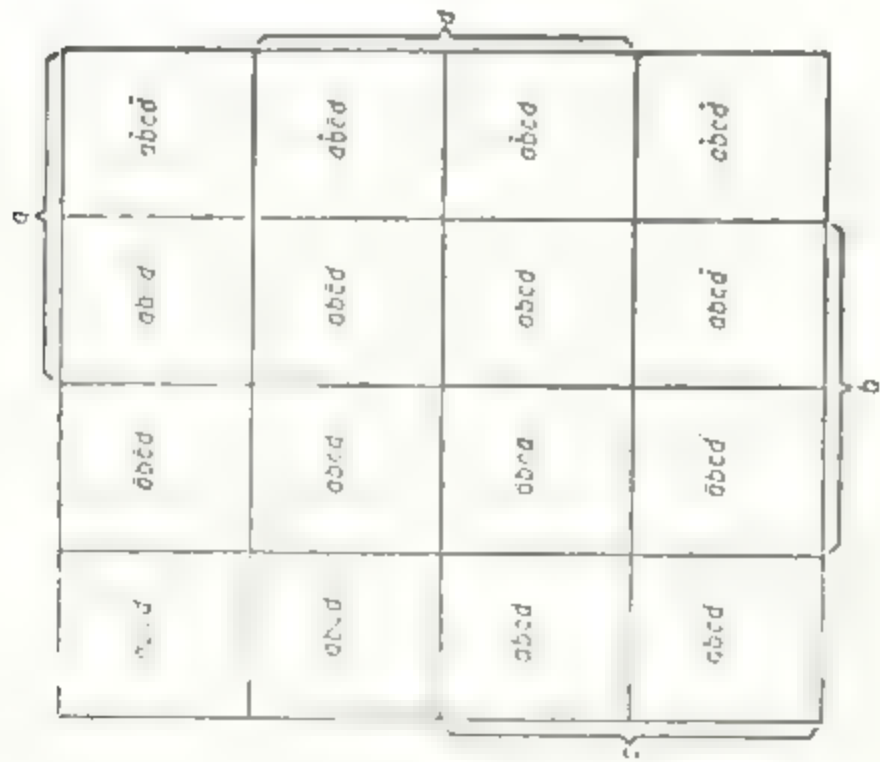
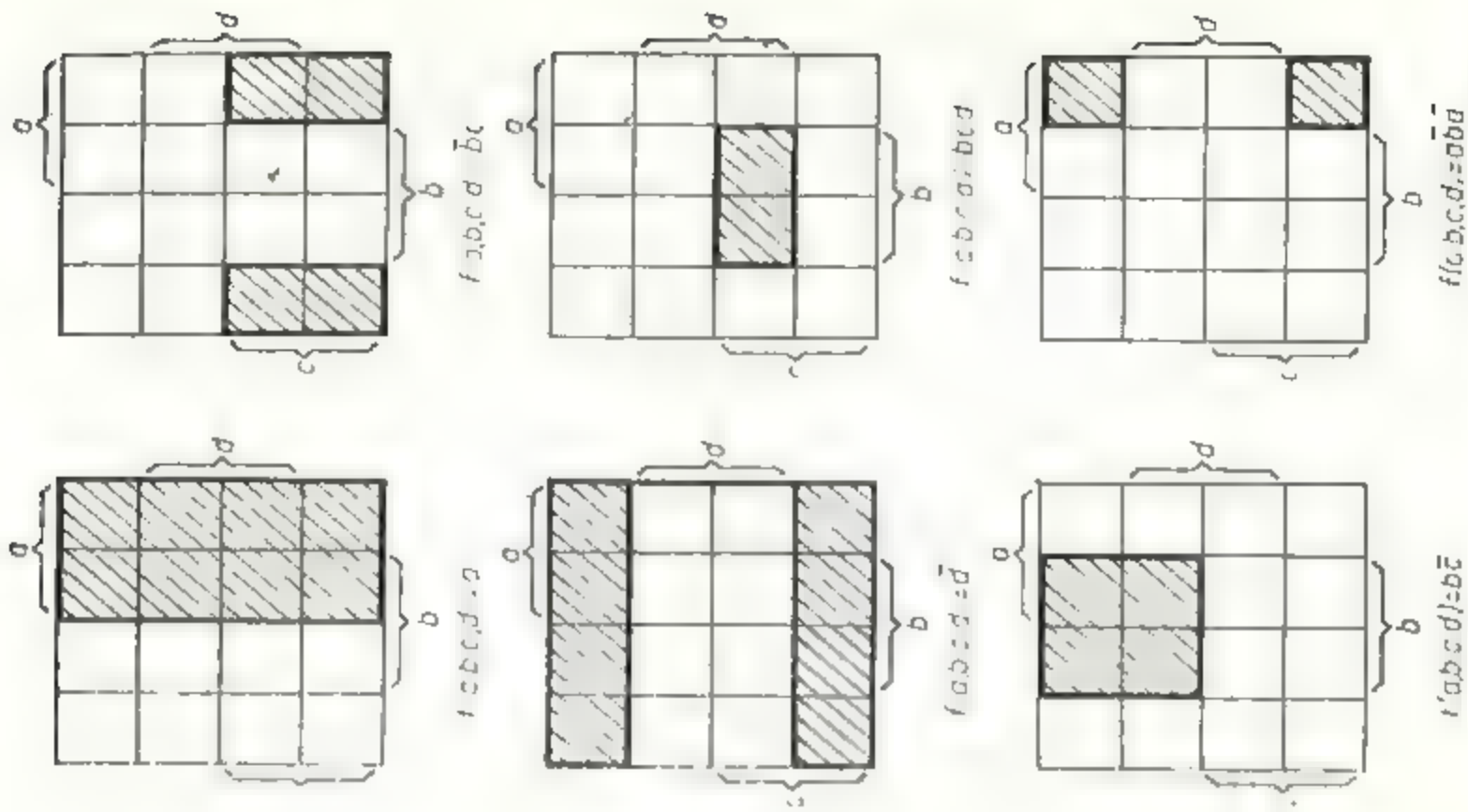


Fig 112



prin pătrate alăturate. Pentru aceasta trebuie însă să se considere că laturile opuse ale diagramei sînt confundate (sau suprapuse). Reciproca acestei proprietăți este de asemenea adevărată. Există însă o proprietate mai generală, deoarece și termenii care nu diferă decît prin complementul unei variabile (abc și $\bar{a}bc$, bd și $\bar{b}d$) se reprezintă pe diagramă prin dreptunghiuri adiacente. Orice dreptunghi, în afara celor cu latura de trei unități corespunde unui termen. Reciproca acestei proprietăți nu reprezintă altceva decît faptul că doi termeni care corespund la două dreptunghiuri alăturate egale pot fi reduși la un singur termen, care corespunde dreptunghiului format prin alăturarea celor două dreptunghiuri. Această proprietate stă la baza metodei de simplificare a funcțiilor booleene cu ajutorul diagramelor Veitch-Karnaugh.

Să considerăm ca exemplu funcția definită prin tabelul I.12, a carei formă canonică disjunctivă este

$$f(a, b, c, d) = \bar{a}b\bar{c}d + \bar{a}bcd + a\bar{b}\bar{c}d + a\bar{b}cd + ab\bar{c}d + abcd + abcd.$$

a	b	c	d	$f(a, b, c, d)$
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Tabelul I.12

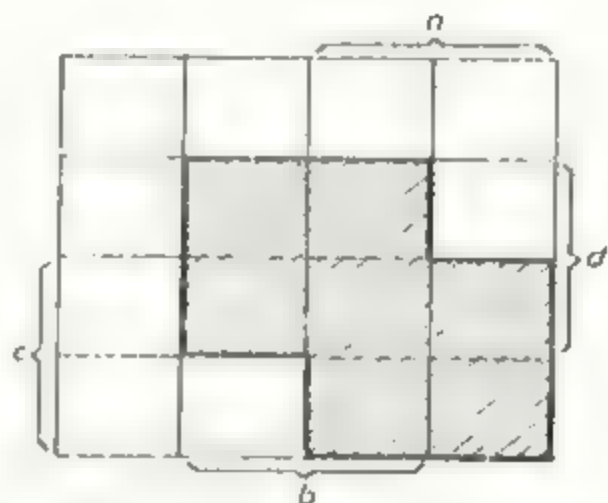


Fig. I.13

În figura I.13 este prezentată diagrama Veitch-Karnaugh corespunzătoare acestei funcții.

Simplificarea funcției corespunde cu acoperirea mulțimii corespunzătoare expresiei respective cu un număr minim de dreptunghiuri cît mai mari. În figura I.14 sînt prezentate trei posibilități de acoperire, corespunzătoare expresiilor $ac + bd$, $ac + b\bar{c}d + \bar{a}bcd$, $b\bar{c}d + \bar{a}bd + \bar{a}bc + acd + abcd$. Se observă că prima expresie, care corespunde acoperirii cu două pătrate mai mari, este cea mai simplă.

Diagrama Veitch-Karnaugh poate fi considerată și ca un tabel de valori cu două dimensiuni al unei funcții. Pentru aceasta, diagrama se rescrie ca în figura I.15. Valorile funcției se vor trece în căsuțele diagramei, considerînd valorile variabilelor funcției drept „coordonate” pentru stabilirea poziției în tabel. Funcția din exemplul anterior va avea diagrama prezentată în figura I.16. Se observă că această diagramă este asemănătoare cu diagrama

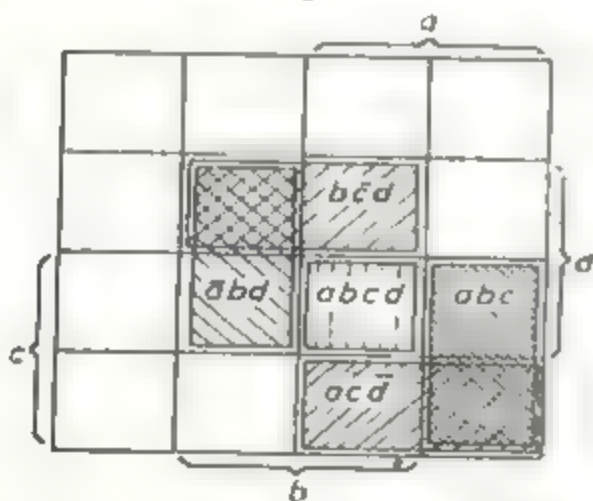
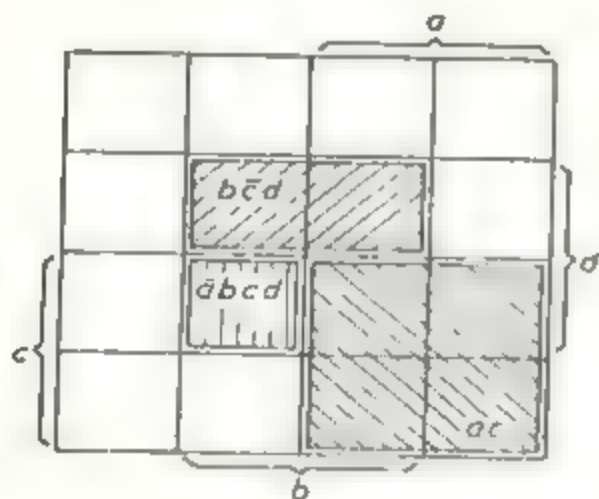
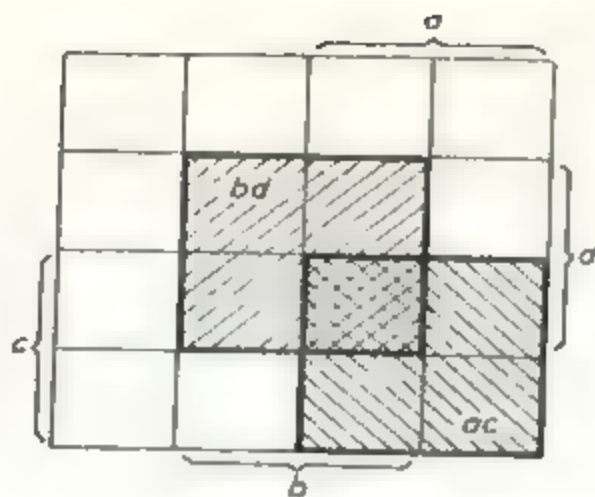


Fig. 1.14

$ab \backslash cd$	00	01	11	10
00				
01				
11				
10				

Fig. 1.15

$ab \backslash cd$	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	0	1	1	1
10	0	0	1	1

Fig. 1.16

clasică. Avantajul acestei forme a diagramelor Veitch-Karnaugh este că deși reprezintă un tabel de valori, poate servi în același timp la simplificarea funcției.

Diagramele Veitch-Karnaugh se utilizează de obicei pentru funcțiile booleene cu maximum cinci variabile (foarte rar pentru șase variabile).

Diagramele Veitch-Karnaugh pentru o funcție cu cinci variabile arată ca în figura 1.17.

Pentru a putea efectua simplificări pe această diagramă se consideră că patratele simetrice față de axa de simetrie a diagramei sînt adiacente. În figura 1.17 funcția reprezentată este :

$$f(a, b, c, d, e) = bc + acd + \bar{a}bc(d + e).$$

ob cd	e = 0				e = 1			
	00	01	11	10	10	11	01	00
00	1	0	0	0	0	0	0	1
01	0	0	1	1	1	1	0	1
11	0	1	1	0	0	1	1	0
10	0	1	1	0	0	1	1	0

Fig 1.17

1.4.3. Exerciții

- Să se simplifice prin calcul boolean expresiile
 - $(a\bar{b} + b)(\bar{a} + \bar{b})$; b) $x_2(\bar{x}_3 + x_4) + x_1x_2(\bar{x}_3 + x_4)$;
 - $x + \bar{z} + y(\bar{x} + \bar{z}) + (x + \bar{z})(x + y + \bar{z})$; d) $\bar{a} + \bar{b} + abc$.
- Sa se simplifice prin calcul boolean complementara expresiilor :
 - $a + b + \bar{c}\bar{d}$; b) $\bar{x}y + z$; c) $\overline{x_1x_2 + \bar{x}_3 + \bar{x}_4}$;
 - $a + b(\bar{c} + \bar{d})$; e) $x_1(x_2x_3 + \bar{x}_4) + x_1(\bar{x}_2\bar{x}_3 + x_4)$.
- Să se găsească prin calcul boolean o formă mai simplă pentru expresiile următoarelor funcții
 - $f(x, y, z) = \bar{x}y\bar{z} + \bar{x}y\bar{z} + \bar{x}yz$;
 - $g(a, b, c) = (\bar{a} + b + \bar{c})(a + \bar{b} + c)(\bar{a} + \bar{b} + \bar{c})$;
 - $h(x_1, x_2, x_3) = x_1x_3 + \bar{x}_1x_2 + x_2x_3$.
- Să se găsească prin calcul boolean o formă normală disjunctivă mai simplă pentru funcțiile
 - $f(x, y, z) = xyz + xy\bar{z} + \bar{x}y + \bar{y}$;
 - $g(x_1, x_2, x_3) = (x_1x_2 + x_1\bar{x}_3)(x_2x_3 + x_1x_3)$;
 - $f(a, b, c) = (a + b)(\bar{a} + \bar{b})(\bar{a} + \bar{b})$.
- Folosind artificii de calcul boolean, să se simplifice expresiile .
 - $ac + b\bar{c} + ab$;
 - $\bar{a}b\bar{c}\bar{d} + ab\bar{c}\bar{d} + \bar{a}b\bar{c}\bar{d} + ab\bar{c}\bar{d}$;
 - $xy + x\bar{y}z + y(\bar{x} + z) + \bar{y}\bar{z}$.
- Funcția majoritară $M(x, y, z)$ se definește ca fiind funcția booleană care are valoarea 1 dacă cel puțin două dintre argumentele sale sînt egale cu 1.
 - Să se alcătuiască tabelul de valori al acestei funcții.
 - Să se afle forma canonică disjunctivă și forma canonică conjunctivă a acestei funcții.
 - Să se determine prin calcul boolean cea mai simplă formă normală disjunctivă
 - Să se arate că $M[a, b, M(c, d, e)] = M[M(a, b, c), d, M(a, b, e)]$.

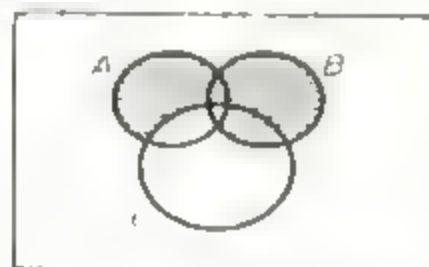
7. Pentru funcțiile din următorul tabel să se afle:

- forma canonică disjunctivă;
- forma canonică conjunctivă;
- cea mai simplă formă normală disjunctivă.

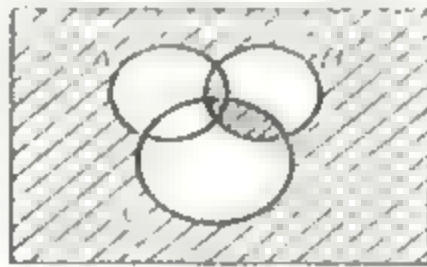
x_1	x_2	x_3	f_1	f_2	f_3
0	0	0	1	1	0
0	0	1	0	1	1
0	1	0	0	0	0
0	1	1	1	1	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	0	1	1
1	1	1	1	1	0

Tabelul 1.13

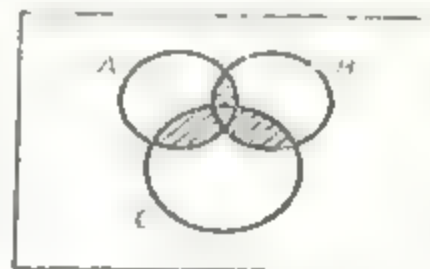
8. Să se indice pentru fiecare din funcțiile ale căror diagrame Euler-Venn sînt reprezentate în figura 1.18 forma canonică disjunctivă și să se simplifice.



a)



b)



c)

Fig. 1.18

9. Să se alcătuiască diagramele Euler-Venn pentru următoarele funcții de trei variabile date prin dezvoltările lor și să se simplifice.

- $m_1 \cup m_2 \cup m_3 \cup m_7$; b) $m_1 \cup m_2 \cup m_4 \cup m_6$;
- $m_6 \cup m_7$; d) $m_1 \cup m_2 \cup m_3 \cup m_4 \cup m_5 \cup m_6$.

unde m_0, m_1, \dots, m_7 corespund notațiilor din figura 1.9.

10. Să se construiască diagramele Veitch Karnaugh pentru următoarele funcții booleene

- $f(a, b, c) = a + b\bar{c} + ac + abc$;
- $f(x_1, x_2, x_3, x_4) = x_1\bar{x}_2x_3 + x_2\bar{x}_3 + \bar{x}_1\bar{x}_2x_4$;
- $f(x, y, z) = x + y + xy + yz + xz + xyz$;
- $g(x, y, z, w) = x\bar{y} + x\bar{z} + yz + \bar{x}\bar{y}$;
- $f(a, b, c, d) = \bar{a}b\bar{c}\bar{d} + \bar{a}b\bar{c}d + \bar{a}bcd + \bar{a}b\bar{c}d + abcd + ab\bar{c}d + abcd$;
- $g(x_1, x_2, x_3, x_4) = (x_1 + \bar{x}_2)x_3 + (x_2 + \bar{x}_4)x_1$.

11. Aflați funcțiile reprezentate prin diagramele Veitch-Karnaugh din figura 1.19.

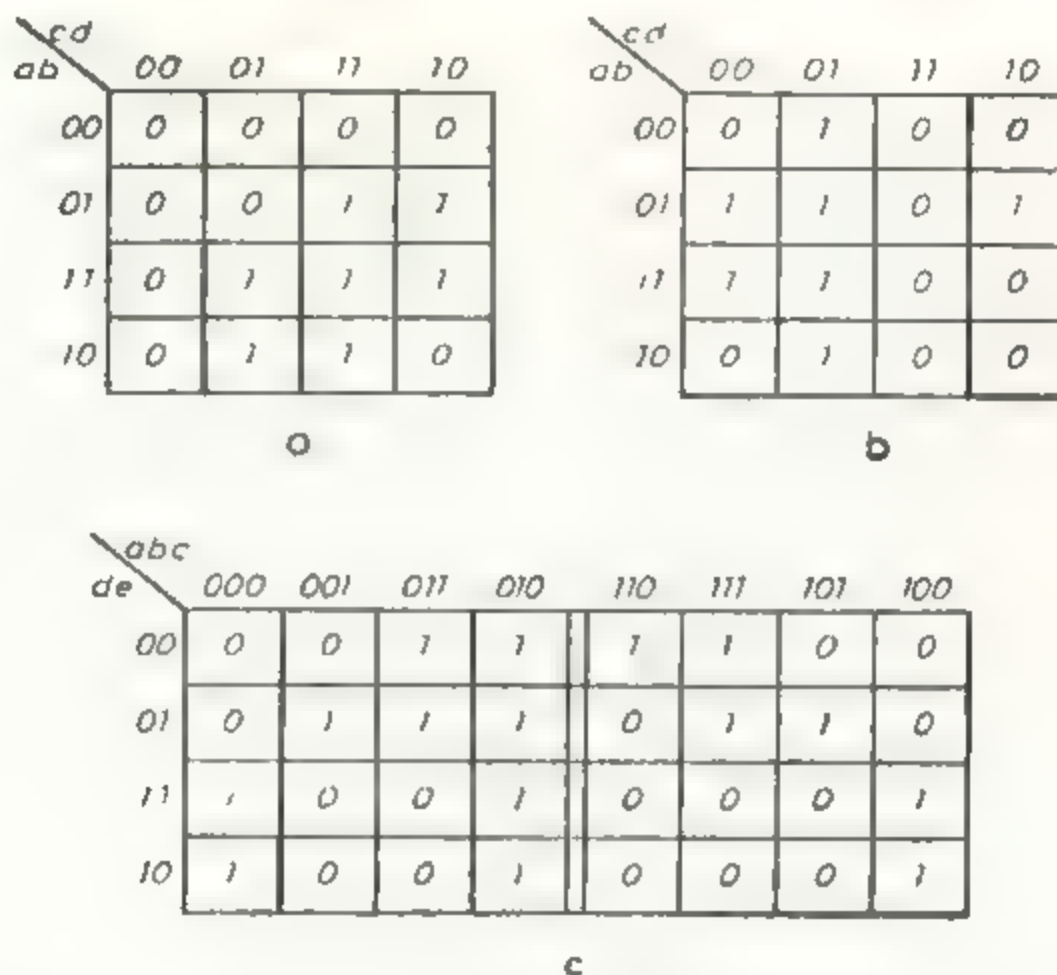


Fig. 1.19

12. Folosind diagrama Veitch-Karnaugh, sa se simplifice următoarele funcții:

- $f(a, b, c, d) = \bar{a}\bar{b} + \bar{a}\bar{c}d + c + b\bar{c}d$;
- $f(x, y, z) = (x + \bar{y} + \bar{z})(\bar{x} + y)(\bar{y} + z)$;
- $g(x, y, z, w) = \bar{x}yz + x\bar{y}w + x\bar{y}z\bar{w} + x\bar{z}w + y\bar{z}w + \bar{x}y\bar{z}w$;
- $h(x_1, x_2, x_3, x_4) = x_2\bar{x}_3\bar{x}_4 + x_1x_2\bar{x}_4 + \bar{x}_1x_3x_4\bar{x}_4$;
- $f(a, b, c, d) = \bar{a}\bar{c}d + \bar{a}b\bar{c}d + \bar{a}b\bar{c} + \bar{a}bd + bcd$.

1.5. Realizarea fizică a funcțiilor booleene

1.5.1. Circuite cu contacte

O prima aplicație a teoriei funcțiilor booleene este reprezentată de studiul circuitelor dipolare cu contacte. Aceste circuite sînt formate prin legarea în serie sau în paralel a unor contacte care sînt de două tipuri: contacte normal deschise și contacte normal închise.

Fiecare contact poate fi pus în corespondență cu o variabilă booleană. Contactele normal închise sînt puse în corespondență cu complementele variabilelor booleene. Un contact se poate afla în două stări: starea de

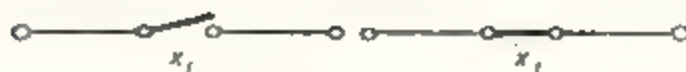


Fig. 1.20

repaus sau normala și starea de lucru. Aceste stări corespund, respectiv, valorilor 0 și 1 ale variabilelor booleene asociate contactului. Un contact normal deschis permite trecerea curentului numai în starea de lucru. Un contact normal închis permite, însă, trecerea curentului numai în starea de repaus. Un circuit dipolar cu contacte nu se poate afla decât în două stări: permite trecerea curentului sau nu o permite. Aceste două stări vor fi asociate cu valorile booleene 1 și respectiv 0.

Considerând cele două circuite simple din figura 1.20 se poate observa că acestea pot fi asociate celor două funcții de o variabilă $f(x_1) = x_1$ și $g(x_1) = \bar{x}_1$. Într-adevar, dacă contactul normal deschis se afla în starea de repaus ($x_1 = 0$), atunci circuitul nu permite trecerea curentului ($f(0) = 0$), iar dacă contactul normal deschis se afla în starea de lucru ($x_1 = 1$) atunci circuitul permite trecerea curentului ($f(1) = 1$). De asemenea, dacă contactul normal închis se afla în starea de repaus ($x_1 = 0$) atunci circuitul permite trecerea curentului ($g(0) = 1$), iar dacă contactul respectiv se afla în starea de lucru ($x_1 = 1$), atunci circuitul nu va permite trecerea curentului ($g(1) = 0$).

Se poate observa că dacă un circuit este obținut prin legarea în serie a altor două circuite mai simple, acestuia i se poate asocia o funcție care corespunde produsului boolean al funcțiilor asociate celor două circuite.

În mod analog funcția asociată unui circuit format prin legarea în paralel a altor două circuite este suma booleană a funcțiilor asociate celor două circuite legate în paralel.

Rezultă, deci, că oricărui circuit dipolar i se poate asocia o funcție booleană care să reprezinte funcționarea circuitului. Trebuie remarcat că, într-un circuit dipolar cu contacte, mai multe contacte pot corespunde aceleiași variabile sau complementului acesteia.

De exemplu, circuitului din figura 1.21 îi corespunde funcția $f(x, y) = x + x\bar{y}$, iar celui din figura 1.22 i se asociază funcția $f(x_1, x_2, x_3, x_4) = (x_1 + x_2)(x_1x_3 + (\bar{x}_1 + x_3)x_4)$.



Fig. 1.21

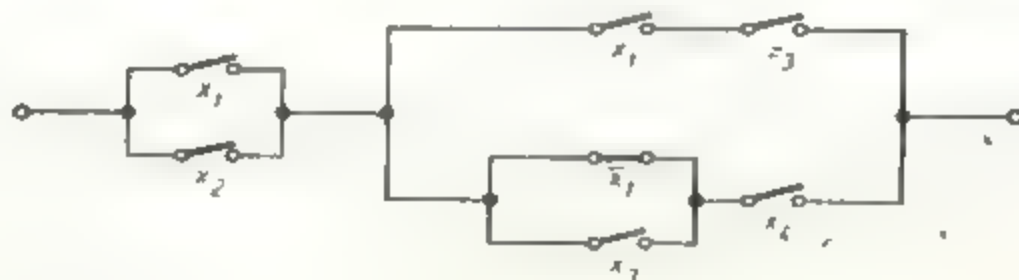


Fig. 1.22

Reciproc, orice funcție booleană poate fi realizată printr-un circuit dipolar cu contacte, adică se poate alcătui un circuit dipolar cu contacte a cărui funcționare să fie reprezentată de funcția dată.

Realizarea funcțiilor booleene prin circuite dipolare cu contacte reprezintă o modalitate de utilizare practică a teoriei funcțiilor booleene.

Este cunoscut că funcționarea celor mai multe circuite de automatizare, precum și funcționarea calculatoarelor electronice se bazează pe utilizarea numerelor binare, care pot fi reprezentate cu numai două cifre, 0 și 1. Funcțiile care reprezintă comportarea în funcționare a circuitelor respective sînt funcții booleene. Aceste circuite pot fi deci construite, realizînd fizic funcțiile booleene corespunzătoare.

Simplificarea unei funcții booleene reprezintă în acest caz realizarea aceleiași funcții printr-un număr cit mai mic de contacte.

În practică, circuitele cu contacte sînt folosite de obicei sub formă de circuite cu rele și contacte. Relele sînt necesare în cadrul acestor circuite pentru a acționa contactele. Un releu va acționa toate contactele corespunzătoare unei variabile, astfel că valorile variabilei pot fi asociate cu cele două stări, de conducere a curentului sau nu, corespunzătoare circuitului de comandă a releului. Se da astfel posibilitatea de a se reprezenta în mod omogen valorile 1 și 0 prin starea de conducere sau nu a curentului prin diferite circuite. Circuitele cu rele și contacte au fost înlocuite treptat prin circuite logice alcătuite din tuburi electronice, apoi prin circuite alcătuite din tranzistoare, iar în ultimul timp prin circuite integrate.

1.5.2. Circuite logice simple

Circuitele logice sînt realizate sub forma unor circuite multipolare (cu mai multe borne). Un circuit logic are una sau mai multe borne de intrare și una sau mai multe borne de ieșire. La bornele de intrare se aplică semnale (în general impulsuri de tensiune electrică), care reprezintă variabile booleene și pot lua două valori (de exemplu, o tensiune de 4 V poate reprezenta valoarea logică „1”, iar o tensiune de 0 V valoarea logică „0”). La bornele de ieșire apar semnale, care corespund valorilor unor funcții booleene de variabilele de la intrare. Aceste circuite logice se vor numi combinaționale. Există posibilitatea ca funcțiile booleene să nu depindă numai de valorile actuale ale variabilelor de la intrare ci și de valorile precedente ale acestor variabile. În acest caz circuitele logice se vor numi secvențiale sau „cu memorie”.

În manualul de față nu vom studia decît circuitele logice combinaționale.

În cazul general, un circuit combinațional are n borne de intrare, corespunzătoare unor variabile booleene x_1, x_2, \dots, x_n și m borne de ieșire, corespunzătoare unor funcții $y_1(x_1, \dots, x_n), \dots, y_m(x_1, \dots, x_n)$. Un astfel de circuit se reprezintă ca în figura 1.23, fără a se descrie structura sa internă.

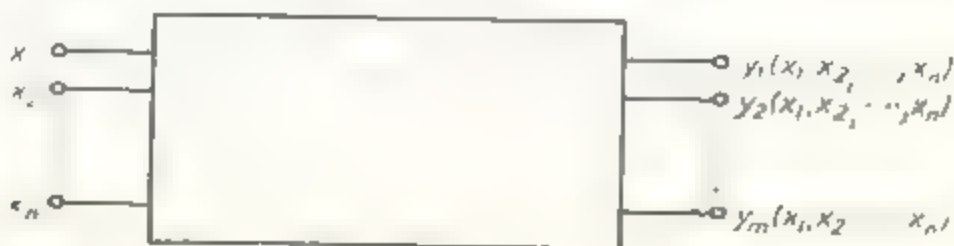


Fig. 1.23



Fig. 1.24

O proprietate importantă a circuitelor logice este că acestea pot fi combinate pentru a forma circuite logice mai complexe. Astfel, bornele de ieșire ale unor circuite pot fi cuplate la bornele de intrare ale altor circuite. Aceasta proprietate permite ca un circuit logic mai complex să poată fi realizat din circuite logice mai simple.

Un circuit combinațional cu mai multe borne de ieșire poate fi conceput ca fiind format din mai multe circuite care au fiecare câte o singură bornă de ieșire și care corespund fiecare la câte o funcție. Un circuit combinațional cu o singură bornă de ieșire va fi denumit circuit logic simplu (fig. 1.24). Funcționarea unui circuit logic simplu este descrisă printr-o funcție booleană de variabilele de la intrare. Putem spune astfel că un circuit logic simplu realizează funcția booleană corespunzătoare. Dacă funcția booleană este o funcție elementară de două variabile, circuitul logic simplu corespunzător care o realizează va fi denumit circuit logic elementar. Există deci trei circuite logice elementare, corespunzătoare celor trei operații de bază ale algebrelor booleene, și anume circuitul ȘI pentru operația „ \cdot ”, circuitul SAU pentru operația „ $+$ ” și circuitul NU pentru operația de complementare. Aceste circuite logice elementare au o reprezentare convențională specială care este aratăată în figura 1.25.

Deoarece orice funcție booleană poate fi reprezentată cu ajutorul celor trei funcții elementare, corespunzătoare celor trei operații „ \cdot ”, „ $+$ ” și „ $\bar{}$ ”, rezultă că orice circuit logic simplu poate fi realizat prin combinarea unor circuite logice elementare. Reprezentarea unei funcții booleene cu ajutorul funcțiilor elementare poate fi considerată drept schema de combinare a circuitelor logice elementare.

De exemplu, funcția

$$f(x_1, x_2, x_3) = x_1 x_2 + \bar{x}_2 \bar{x}_3$$

care poate fi reprezentată sub forma

$$((x_1 \cdot x_2) + ((\bar{x}_2) \cdot (\bar{x}_3))),$$

este realizată de circuitul logic simplu din fig. 1.26.

Dacă reprezentăm însă aceeași funcție prin

$$f(x_1, x_2, x_3) = x_1 x_2 + \bar{x}_2 + x_3,$$

realizarea funcției va fi diferită, corespunzând circuitului logic din figura 1.27.

Rezultă deci, că o funcție booleană poate fi realizată de diferite circuite logice simple, corespunzând diferitelor reprezentări ale acesteia. Se poate acum înțelege importanța simplificării funcțiilor booleene, deoarece astfel se pot realiza aceleași funcții, însă cu scheme mai simple și economice.

În 1.2.3 s-a arătat că orice funcție booleană poate fi reprezentată cu ajutorul unei singure funcții de două variabile, această funcție fiind fie funcția lui Sheffer $f_{11}(x, y) = \bar{x} + \bar{y}$, fie funcția lui Peirce $f_1(x, y) = \bar{x}y$. Există circuite logice simple care realizează aceste funcții.

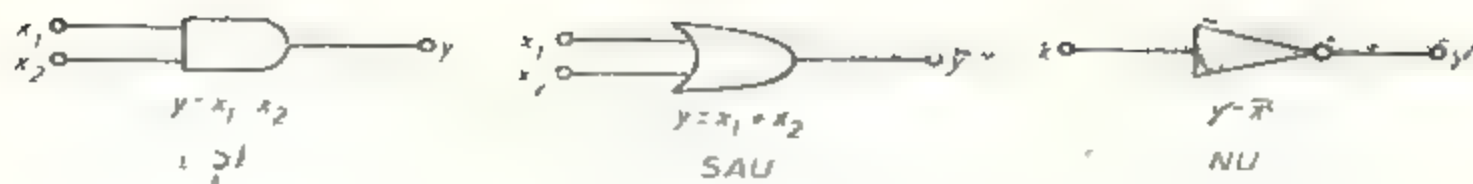


Fig. I.25

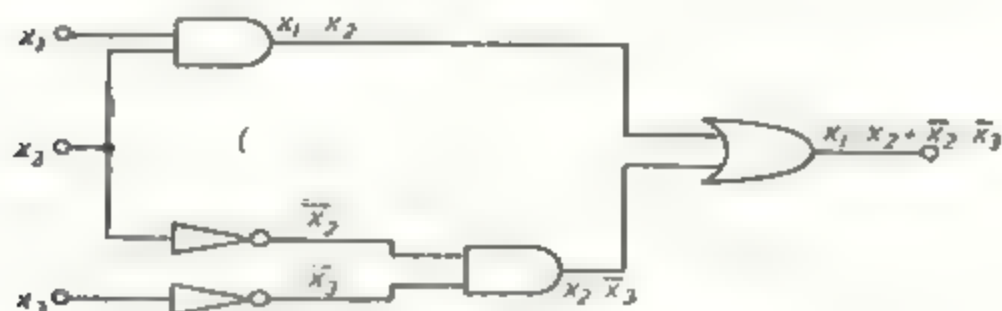


Fig. I.26

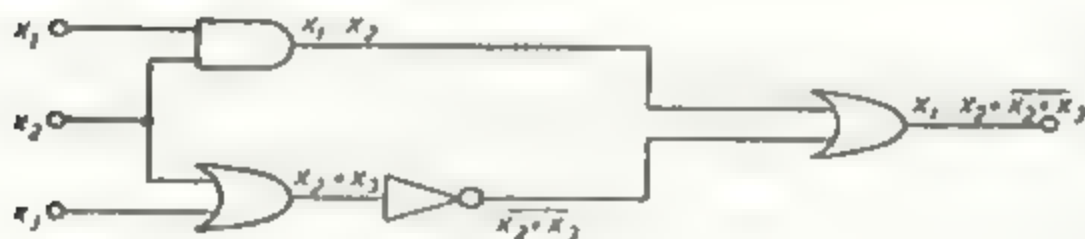


Fig. I.27

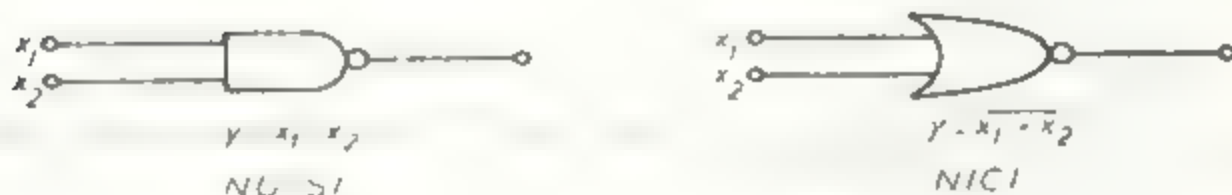


Fig. I.28

Aceste circuite vor fi denumite circuite logice universale. Rezultă, deci, că orice circuit logic simplu poate fi realizat printr-o combinație de circuite logice universale. Circuitele logice universale sînt simbolizate ca în figura I.28. Aceste circuite poartă denumiri proprii, particulare. Astfel circuitul care realizează funcția lui Sheffer se numește circuit NU-ȘI, iar cel care realizează funcția lui Peirce se numește circuitul NICI.

Funcția considerată anterior poate fi reprezentată astfel :

— prin funcția lui Sheffer (notată cu „|“)

$$f(x_1, x_2, x_3) = (x_1 | x_2) | ((x_1 | x_2) | (x_2 | x_3))$$

— prin funcția lui Peirce (notată cu „↓“)

$$g(x_1, x_2, x_3) = ((x_1 \downarrow x_1) \downarrow (x_2 \downarrow x_2)) \downarrow (x_2 \downarrow x_3),$$

$$\text{iar } f(x_1, x_2, x_3) = g(x_1, x_2, x_3) \downarrow g(x_1, x_1, x_3).$$

Circuitele logice simple corespunzătoare acestor reprezentări pot fi văzute în figura I.29 și figura I.30.

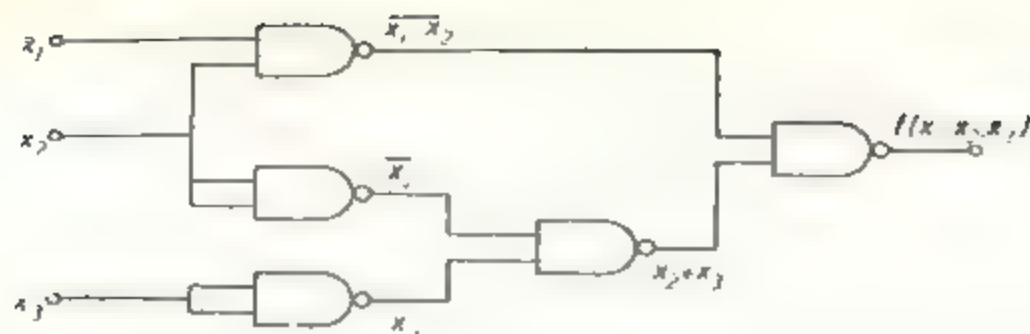
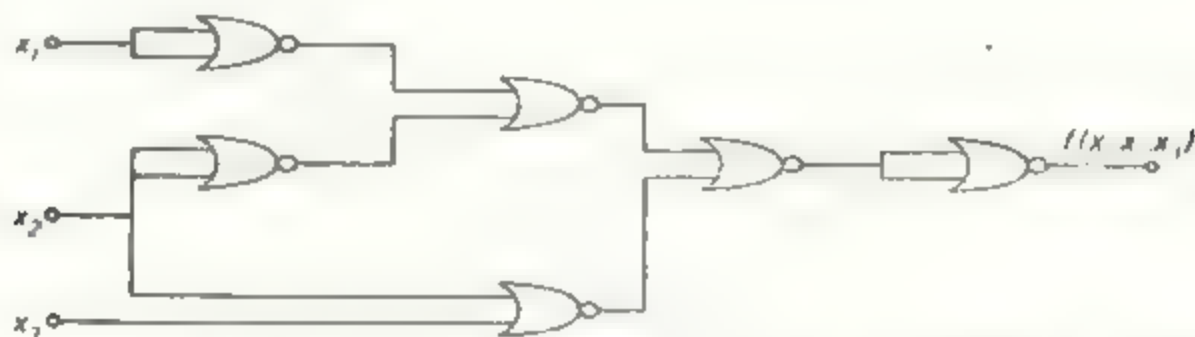


Fig. 1.29

Fig. 1.30



Deși schemele acestor circuite sînt mai complicate decît cele ale circuitelor din figurile 1.26 și 1.27, compuse din circuite logice elementare, sînt totuși mai ușor de realizat din punct de vedere tehnic, deoarece sînt alcătuite din același tip de circuite. De obicei, însăși circuitele logice universale sînt mai ușor de realizat și conțin mai puține piese, iar uneori este mai economic să se realizeze chiar și circuitele logice elementare prin combinații ale circuitelor logice universale.

Realizarea circuitelor logice elementare prin circuite NU ȘI este prezentată în figura 1.31 și are la bază următoarele reprezentări:

$$x_1 \cdot x_2 = \overline{\overline{x_1 x_2} \cdot \overline{x_1 x_2}} = (x_1 \downarrow x_2) \downarrow (x_1 \downarrow x_2),$$

$$x_1 + x_2 = \overline{\overline{x_1 x_1} \cdot \overline{x_2 x_2}} = (x_1 \downarrow x_1) \downarrow (x_2 \downarrow x_2),$$

$$\overline{x_1} = \overline{x_1 + x_1} = (x_1 \downarrow x_1).$$

Realizarea circuitelor logice elementare prin circuite NICI este prezentată în figura 1.32 și are la bază următoarele reprezentări:

$$x_1 \cdot x_2 = \overline{\overline{x_1 + x_1 + x_2 + x_2}} = (x_1 \downarrow x_1) \downarrow (x_2 \downarrow x_2),$$

$$x_1 + x_2 = \overline{\overline{x_1 + x_2 + x_1 + x_2}} = (x_1 \downarrow x_2) \downarrow (x_1 \downarrow x_2),$$

$$\overline{x_1} = \overline{x_1 + x_1} = x_1 \downarrow x_1.$$

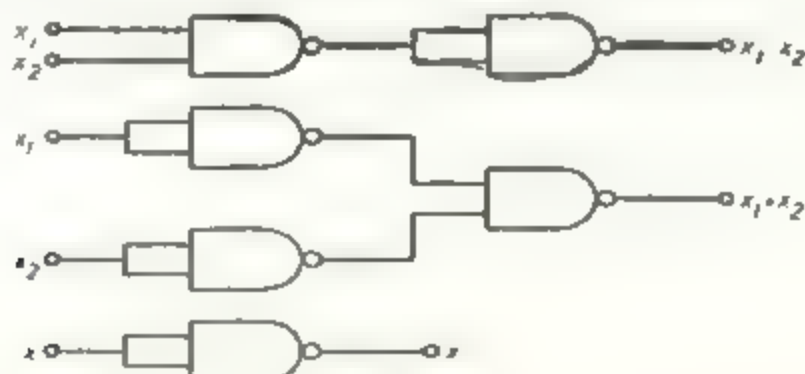


Fig. 1.31

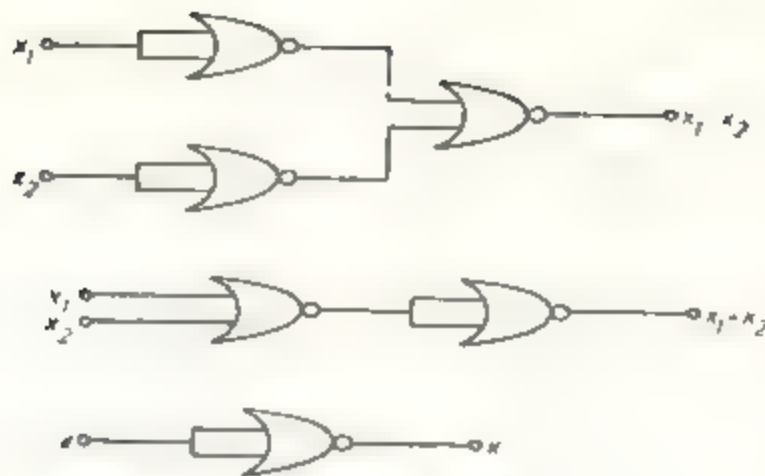


Fig. 1.32

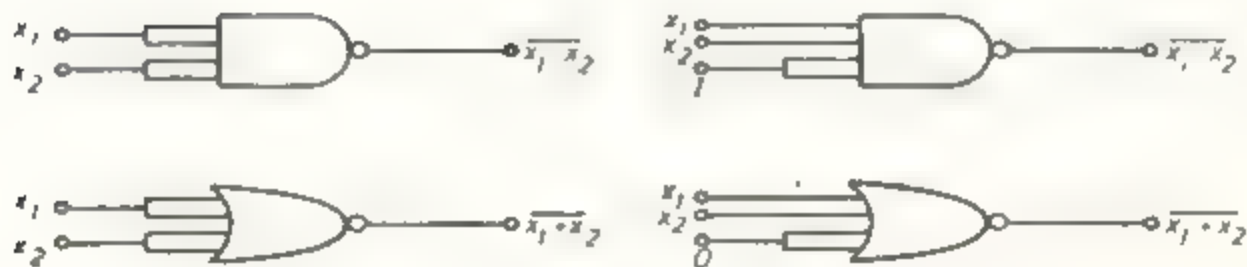


Fig. 1.33

În tehnică sînt de asemenea ușor de realizat circuite ȘI, SAU, NICI sau NU-ȘI cu mai mult decît două borne de intrare. Aceste circuite au aceleași simboluri ca și circuitele corespunzătoare cu numai două borne de intrare. Un astfel de circuit poate fi realizat oricînd în locul unui circuit similar cu mai puține borne de intrare, dacă la bornele de intrare suplimentare se aplică fie semnale avînd o valoare logică constantă („1” pentru circuitele „ȘI” și „NU-ȘI” și „0” pentru „SAU” și „NICI”), fie unul dintre semnalele de la celelalte borne. În figura 1.33 se prezintă exemple de utilizare a unui circuit cu patru borne de intrare pentru realizarea unei funcții booleene cu două variabile.

Circuitele logice simple cu un număr mai mare de borne de intrare de tip ȘI și SAU pot fi utilizate la realizarea funcțiilor booleene reprezentate sub o formă normală.

De exemplu, funcția $f(x_1, x_2, x_3) = x_1x_2\bar{x}_3 + x_1\bar{x}_2x_3 + \bar{x}_1x_2x_3$ poate fi realizată cu circuitul din figura 1.34.

În practică pot apare situații care să necesite aflarea funcției booleene realizate de un circuit logic simplu a cărui schemă (structură) este cunoscută.

Pentru a obține expresia funcției booleene realizate de un circuit se procedează astfel: se identifică variabilele booleene corespunzătoare bornelor de intrare ale circuitului; se alcătuiesc expresiile booleene corespunzătoare acelor circuite elementare pentru care s-au identificat variabilele de la intrare; aceste expresii constituie intrările altor circuite elementare; se obțin astfel expresii din ce în ce mai complexe, iar în final expresia căutată, corespunzătoare ieșirii circuitului.

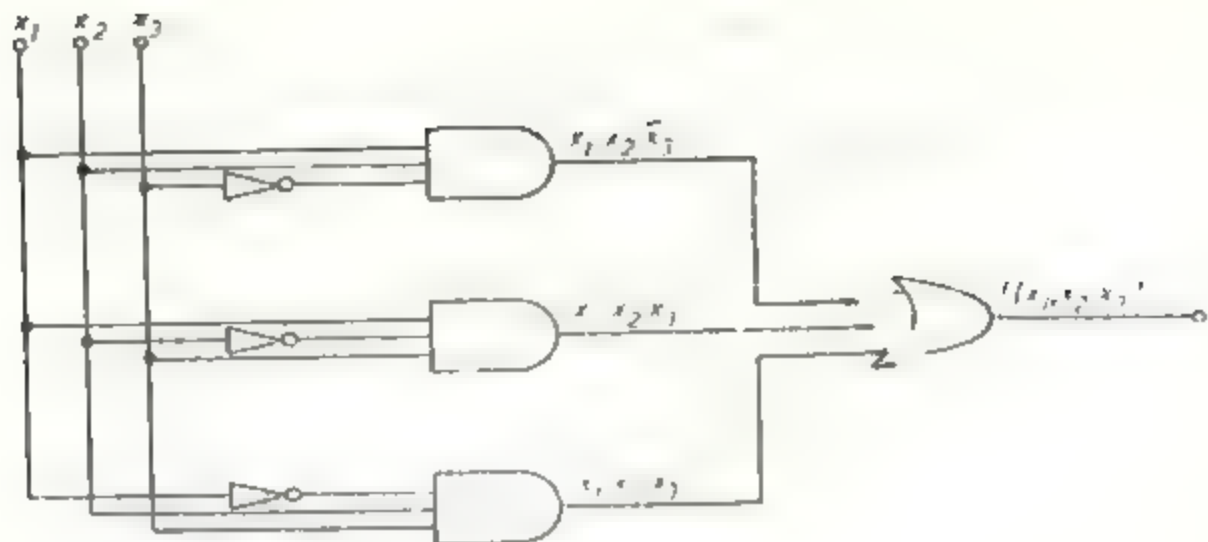


Fig. 1.34

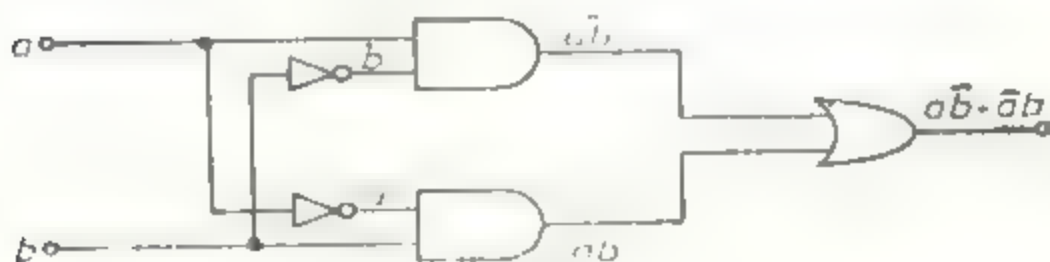


Fig. 1.35

De exemplu, circuitul din figura 1.35 are ca variabile de intrare a și b . Dacă în dreptul ieșirii fiecărui circuit notăm expresiile boolene corespunzătoare, vom scrie pe rând, \bar{a} , \bar{b} , $a\bar{b}$, $\bar{a}b$, $a\bar{b} + \bar{a}b$.

1.5.3. Proiectarea circuitelor logice

Circuitele logice reprezintă o parte importantă din circuitele unui calculator electronic. Ele realizează diferite expresii boolene care sînt necesare pentru funcționarea calculatorului. Este cunoscut că funcționarea calculatorului se bazează pe transformarea unor semnale binare, ale căror valori „0” sau „1” au fiecare o anumită semnificație. De exemplu, un număr se reprezintă întotdeauna în baza 2, astfel că cifrele sale nu sînt decît 0 sau 1. Un număr cu cinci cifre binare se reprezintă cu ajutorul a cinci semnale binare, câte un semnal pentru fiecare cifră. Semnul unui număr poate fi de asemenea reprezentat printr-un semnal binar, considerînd că valoarea 0 corespunde semnelui $+$, iar valoarea 1 semnelui $-$.

Transformarea semnalelor binare este efectuată conform unor reguli bine stabilite, în funcție de rezultatul urmărit. Adunarea a două numere binare se reduce la o transformare a semnalelor corespunzătoare celor doi operanzi, pentru a se obține semnalele corespunzătoare sumei, conform regulilor de adunare a două numere.

O activitate importantă din cadrul etapei de proiectare a unui calculator electronic o constituie proiectarea circuitelor logice din care este alcătuit

Proiectarea unui circuit logic se desfășoară în mai multe faze, descrise pe scurt în continuare:

— stabilirea variabilelor de intrare și de ieșire precum și a semnificației valorilor acestora;

— determinarea funcției booleene (sau a funcțiilor, în cazul mai multor variabile de ieșire), care corespunde transformărilor necesare ale variabilelor de intrare și alcatuirea tabelului de valori corespunzător;

— aflarea unei expresii booleene, de obicei sub forma canonică disjunctivă, corespunzătoare fiecărei funcții;

— aflarea celei mai simple expresii pentru fiecare funcție booleană;

— alcatuirea unui circuit logic care să realizeze funcțiile booleene în forma simplificată.

Este posibil ca unele etape să nu fie necesare în anumite cazuri. De exemplu, dacă tabelul de valori al funcției este alcătuit sub forma de diagrama Veitch-Karnaugh, nu este necesară aflarea unei expresii oarecare a funcției, ci se poate trece direct la aflarea celei mai simple expresii a funcției respective.

Proiectarea unui circuit cu contacte se face urmînd aceleași faze astfel ca în ultima fază să se alcatuiască schema circuitului respectiv.

Pentru exemplificare se va prezenta proiectarea unui circuit comparator.

Circuitul comparator care trebuie proiectat este destinat să compare două numere de câte două cifre, indicînd pe cel mai mare dintre acestea.

Variabilele de intrare sînt în număr de patru. Alegem, de exemplu, ca x_1 și x_2 să reprezinte cifrele binare ale primului număr, iar x_3 și x_4 pe cele ale numărului al doilea. Semnalul de ieșire va indica numărul cel mai mare astfel, dacă primul număr este cel mai mare, semnalul de ieșire va avea valoarea „1”, iar dacă al doilea număr este cel mai mare, semnalul de ieșire va avea valoarea „0”. Dacă cele două numere sînt egale, convenim să considerăm al doilea număr ca fiind cel mai mare. În tabelul 1.14 se indică valorile funcției circuitului comparator considerînd că numerele sînt reprezentate în baza 10, iar în tabelul 1.15 funcția circuitului este definită ca o funcție booleană. Diagrama Veitch-Karnaugh a acestei funcții este prezentată în figura 1.36.

n_1	n_2	f
0	0	0
0	1	0
0	2	0
0	3	0
1	0	1
1	1	0
1	2	0
1	3	0
2	0	1
2	1	1
2	2	0
2	3	0
3	0	1
3	1	1
3	2	1
3	3	0

Tabelul 1.14

x_1	x_2	x_3	x_4	f
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Tabelul 1.15

$x_1 \backslash x_2 \backslash$	0	1	2	3
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0

Fig. 1.36

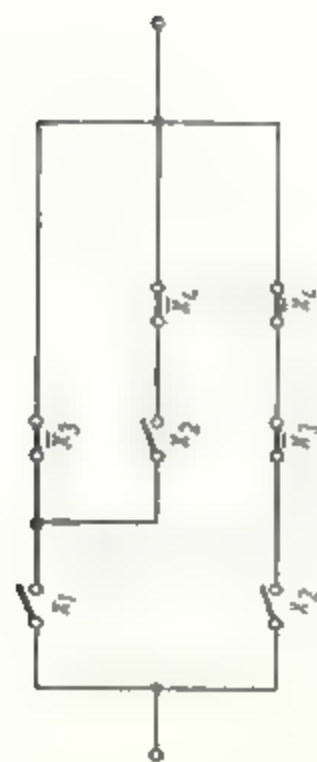


Fig. I.37

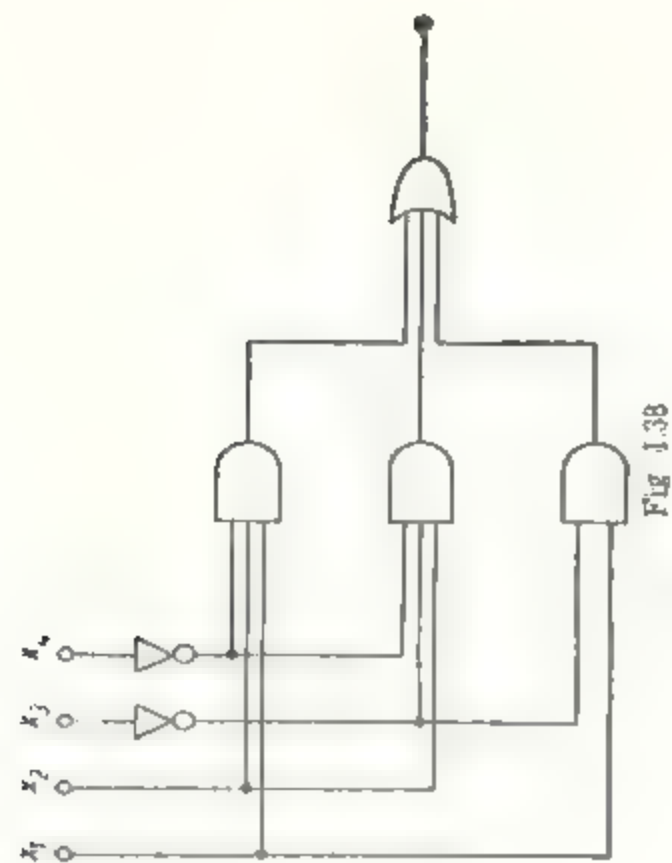


Fig. I.38

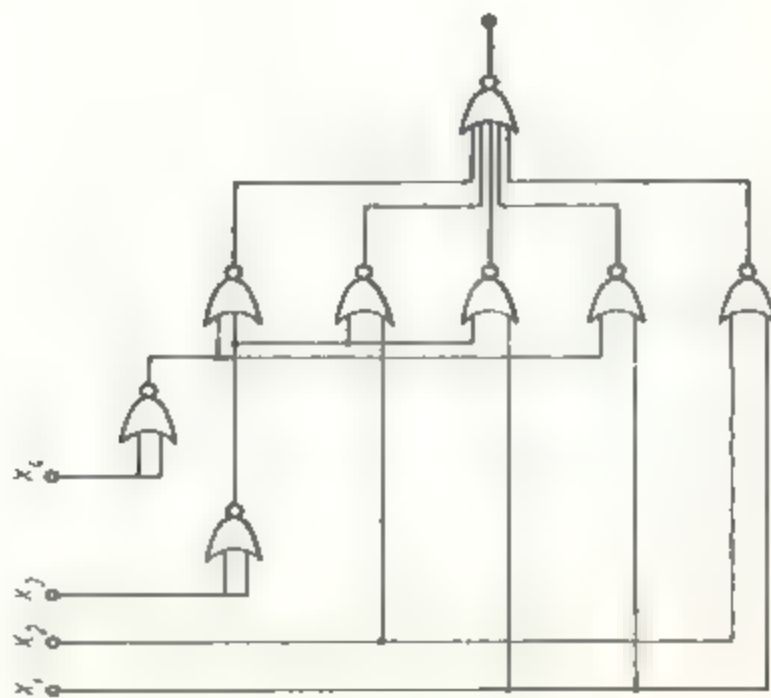


Fig. I.39

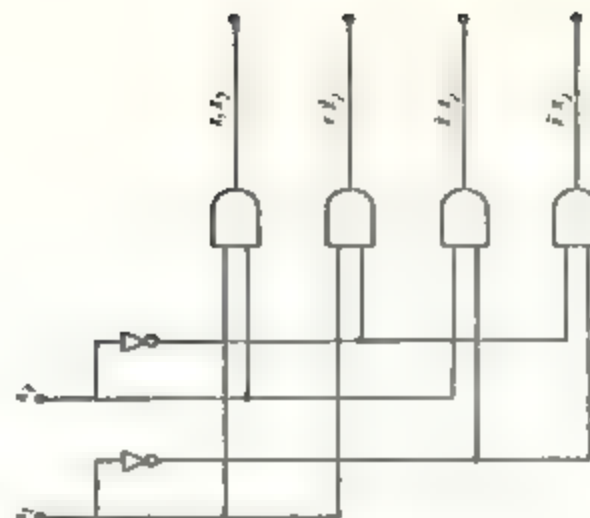


Fig. I.41

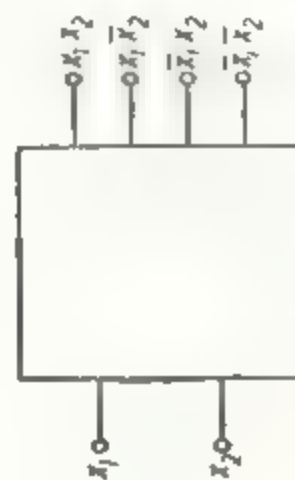


Fig. I.40

Forma canonică disjunctivă a acestei funcții este

$$f(x_1, x_2, x_3, x_4) = \bar{x}_1 x_2 \bar{x}_3 \bar{x}_4 + x_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 + x_1 \bar{x}_2 x_3 x_4 + x_1 x_2 \bar{x}_3 \bar{x}_4 + x_1 x_2 \bar{x}_3 x_4 + x_1 x_2 x_3 \bar{x}_4.$$

Cea mai simplă formă normală disjunctivă este :

$$f(x_1, x_2, x_3, x_4) = x_1 \bar{x}_3 + x_1 x_2 \bar{x}_4 + x_3 \bar{x}_3 \bar{x}_4,$$

dupa cum se poate vedea și pe diagrama din figura I.36. Corespunzător acestei forme normale se obține circuitul cu contacte din figura I.37 și circuitul logic din figura I.38.

Dacă se dorește realizarea circuitului logic numai prin circuite NCI, transformând funcția definită anterior, se obține expresia

$$f(x_1, x_2, x_3, x_4) = \overline{\bar{x}_1 \bar{x}_2 + \bar{x}_1 x_3 + x_1 x_4 + \bar{x}_3 x_3 + x_3 x_4}$$

și circuitul asociat din figura I.39.

1.5.4. Circuite logice complexe

În echipamentele electronice sînt utilizate adeseori circuite logice complexe, care realizează simultan mai multe funcții booleene. Aceste funcții, de obicei nu sînt independente, ci se completează reciproc.

Vom analiza în continuare cîteva circuite logice complexe, mai frecvent utilizate.

Decodificatoare

Circuitele decodificatoare sînt destinate să recunoască diferitele combinații de valori ale variabilelor de la intrare. Schematic, un circuit decodificator se reprezintă ca în figura I.40, indicîndu-se pentru fiecare bornă de ieșire combinația de valori ale variabilelor de intrare corespunzătoare. Rezultă, deci, că funcțiile booleene realizate de un astfel de circuit sînt compuse fiecare dintr-un singur mintermen. Se poate astfel observa că pentru orice combinație de valori ale variabilelor de intrare, una și numai una dintre bornele de ieșire va indica valoarea logică „1” (și anume borna de ieșire a funcției compuse din mintermenul asociat respectivei combinații de valori ale variabilelor). În figura I.41 se prezintă un circuit decodificator pentru două variabile.

În practică pot fi necesare uneori circuite decodificatoare care să indice absența unor combinații de valori ale variabilelor de la intrare. Desigur că în acest caz pentru fiecare combinație de valori ale variabilelor de la intrare numai o singură bornă de ieșire va indica valoarea logică „0” (și anume borna de ieșire a funcției, corespunzătoare acelei combinații, deoarece această combinație nu este absentă). Un exemplu de astfel de circuit decodificator cu două variabile este prezentat în figura I.42.

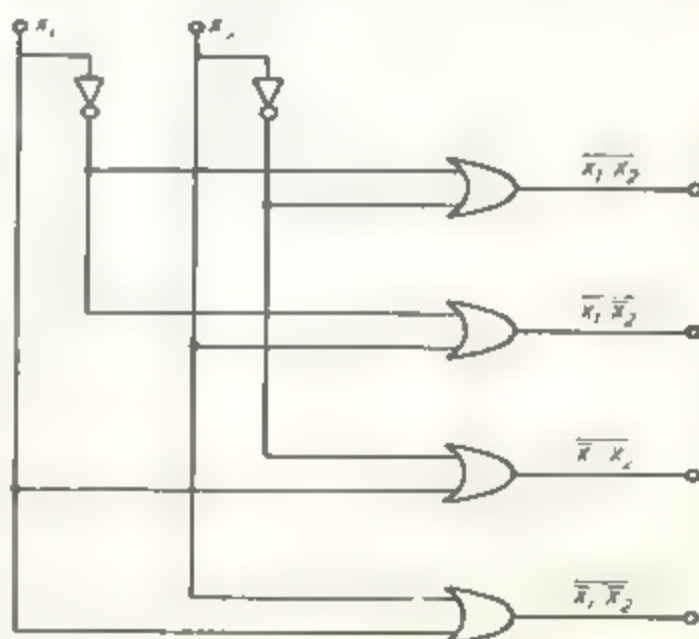


Fig. I 42

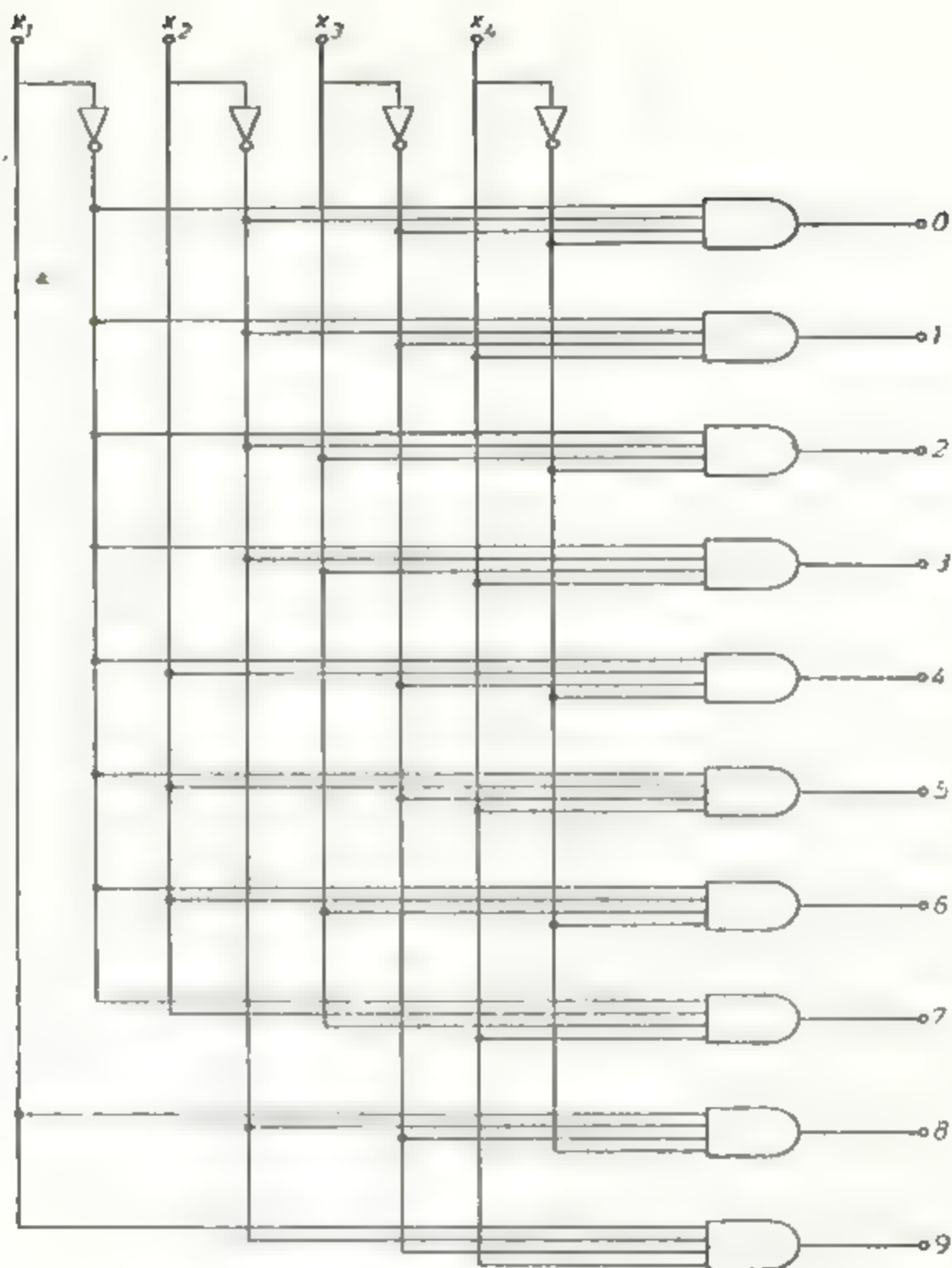


Fig 1.43

Există și circuite decodificatoare reduse, care nu au borne de ieșire pentru toate combinațiile de valori posibile ale variabilelor de intrare, ci numai pentru cele necesare. Un exemplu de astfel de decodificator este prezentat în figura 1.43. Acest circuit servește la decodificarea cifrelor zecimale codificate în binar, conform tabelului 1.16.

x_1	x_2	x_3	x_4	f
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9

Tabelul 1.16

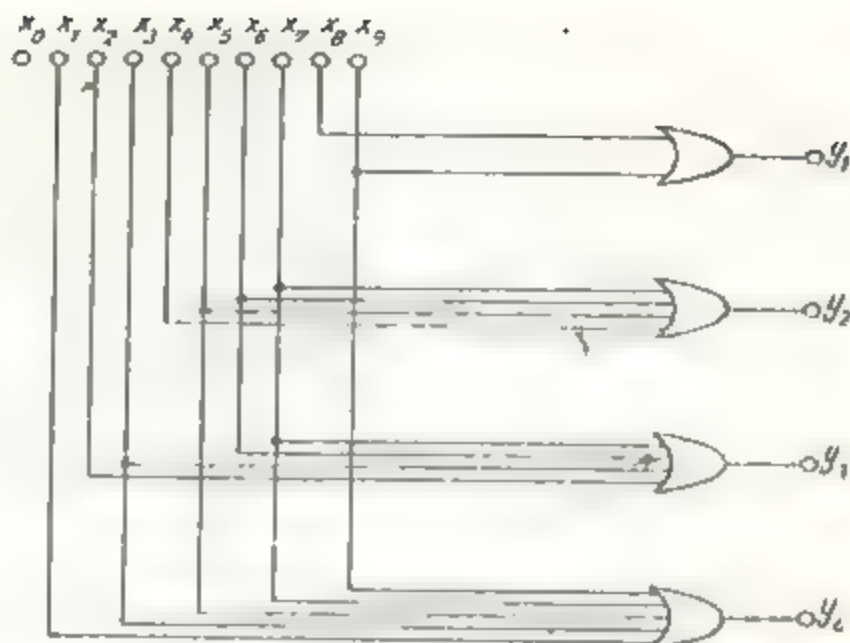


Fig. 1.44

Codificatoare

Circuitele codificatoare realizează o operație inversă în raport cu circuitele decodificatoare. Astfel, variabilele de la bornele de intrare ale acestor circuite nu capătă toate combinațiile de valori posibile, deoarece atunci când o variabilă are valoarea „1”, celelalte variabile nu pot avea decât valoarea „0”. Rezultă, deci, ca numărul de combinații ale variabilelor de intrare este egal cu numărul variabilelor de intrare. Pentru fiecare dintre aceste combinații se obține la bornele de ieșire o combinație de valori care reprezintă codul semnalului de la intrare. Un circuit codificator este compus din mai multe circuite SAU, câte unul pentru fiecare bornă de ieșire. Fiecare circuit SAU are mai multe intrări. În figura 1.44 este prezentat un circuit codificator care codifică în binar cifrele 0, 1, ..., 9. Codul considerat pentru fiecare cifră este indicat în tabelul 1.17. Se observă că borna de intrare corespunzătoare cifrei 0 nu este legată cu alt circuit, deoarece codul necesar este format din valorile 0 pentru toate bornele de ieșire.

Sumatoare

Unul dintre circuitele cele mai importante ale unui calculator electronic îl constituie sumatorul. Acest circuit participă la efectuarea operațiilor aritmetice. Este cunoscut că numerele cu care lucrează un calculator electronic sînt de obicei reprezentate în baza de numerație 2.

Adunarea a două numere binare (în baza 2) se efectuează conform regulilor obișnuite, cifră cu cifră. Un circuit sumator este realizat pe baza circuitului sumator elementar. Circuitul sumator elementar efectuează adunarea a două cifre binare. Rezultatul adunării este alcătuit din două cifre binare: o cifră a sumei și o cifră transport care trebuie adunată la cifrele coloanei alăturate din stînga. Aceasta înseamnă că în realitate un circuit sumator elementar va

	y_1	y_2	y_3	y_4
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Tabelul 1.17

<i>a</i>	<i>b</i>	<i>t</i>	<i>s</i>	<i>t'</i>
0	0	0	0	0
1	0	0	1	0
0	1	0	1	0
1	1	0	0	1
0	0	1	1	0
1	0	1	0	1
0	1	1	0	1
1	1	1	1	1

Tabelul 1.18

trebui să efectueze adunarea a trei cifre binare (cîte o cifră de la cele două numere și cifra transport de la coloana precedentă).

Notînd cu *a* și *b* cifrele celor doi operanzi, cu *t* cifra transport de la coloana anterioară, cu *s* cifra sumei și cu *t'* cifra transport către coloana următoare, se poate alcătui tabelul 1.18.

Putem deci considera *s* și *t'* ca funcții booleene de *a*, *b* și *t*. Diagramele celor două funcții sînt indicate în figura 1.45, iar în figura 1.46 se prezintă o diagramă comună pentru cele două funcții.

Din diagramele Veitch Karnaugh se pot deduce expresiile booleene simplificate ale celor două funcții :

$$s = \bar{a}\bar{b}t + \bar{a}bt + a\bar{b}t + abt \text{ și}$$

$$t' = ab + at + bt.$$

Folosind aceste expresii, un sumator elementar are schema prezentată în figura 1.47.

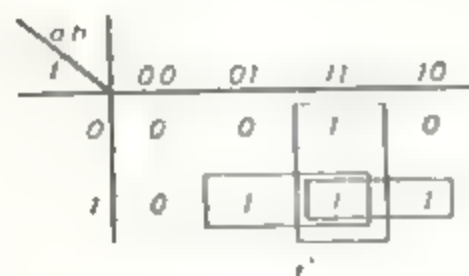
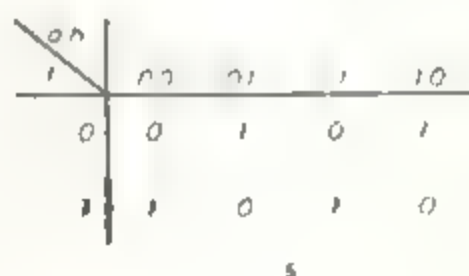


Fig. 1.45

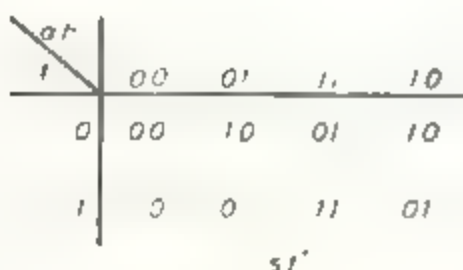


Fig. 1.46

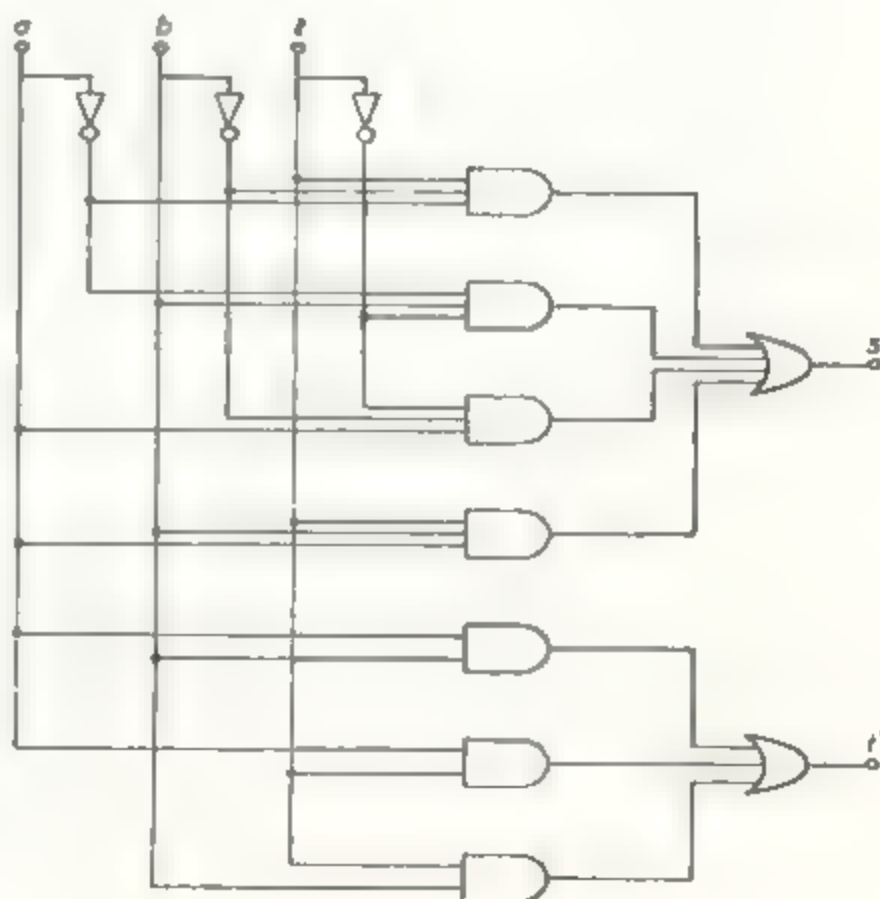


Fig. 1.47

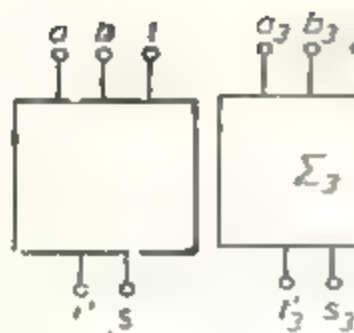


Fig. 1.48

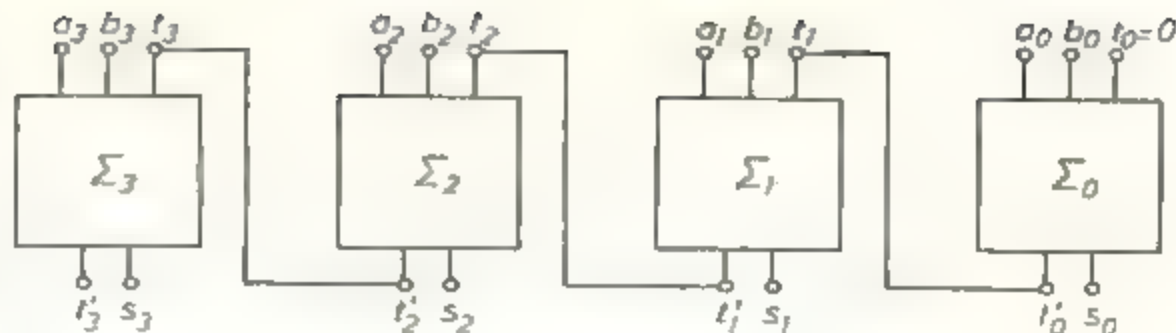


Fig. 1.49

Simbolic, un sumator elementar poate fi reprezentat ca orice circuit logic (fig. 1.48)

Un sumator pentru numere binare cu n cifre este alcătuit din n sumatoare elementare, care adună fiecare câte o cifră. Circuitele sumatoare elementare sînt interconectate pentru a-și transmite reciproc cifrele de transport respective. Schema unui sumator pentru patru cifre binare este prezentată în figura 1.49. Un astfel de sumator este denumit sumator paralel, deoarece cifrele celor două numere sînt introduse simultan.

Dacă este memorată cifra transport, atunci un singur sumator elementar poate fi utilizat (în mod repetat) pentru calculul tuturor cifrelor rezultatului. Un astfel de circuit, denumit sumator serie, nu mai constituie însă un circuit combinațional, ci un circuit secvențial, iar realizarea fizică a sa este mai complicată decît a sumatorului paralel.

1.5.5. Realizarea practică a circuitelor logice

Circuitele logice elementare sînt realizate practic utilizînd elemente de circuit obișnuite: rezistențe, condensatoare, diode, tranzistori. Există diferite scheme de circuite electrice care realizează funcții booleene.

Vom prezenta cîteva exemple de scheme simple care realizează operațiile booleene elementare. În cazul acestor scheme se presupune că valoarea logică 0 este reprezentată de o tensiune de 0 V, iar valoarea logică 1 de o tensiune de +4 V.

În figura 1.50 este prezentată schema unui circuit ȘI. Analizînd funcționarea circuitului, se observă că dacă una dintre bornele de intrare are tensiunea 0 V, dioda corespunzătoare se află în stare de conducție, astfel că borna de ieșire are de asemenea tensiunea 0 V. Deci, borna de ieșire va avea tensiunea de +4 V numai dacă toate bornele de intrare vor avea de asemenea o tensiune de +4 V, toate diodele fiind astfel blocate.

În figura 1.51 este indicată schema unui circuit SAU. În acest caz, oricare dintre bornele de intrare ar avea tensiunea de +4 V, dioda respectivă ar fi în conducție, astfel că la borna de ieșire s-ar obține tensiunea de +4 V. Tensiunea de la borna de ieșire este 0 numai dacă la toate bornele de intrare tensiunea este 0.

Circuitul din figura 1.52 realizează funcția de complementare (circuit logic NU). Conform principiilor de funcționare ale unui tranzistor, dacă tensiunea de intrare este +4 V, tranzistorul se află în stare de conducție, astfel că tensiunea de ieșire este 0 V, iar dacă tensiunea de intrare are valoarea de 0 V, tranzistorul este blocat, astfel că tensiunea de ieșire este de +4 V.

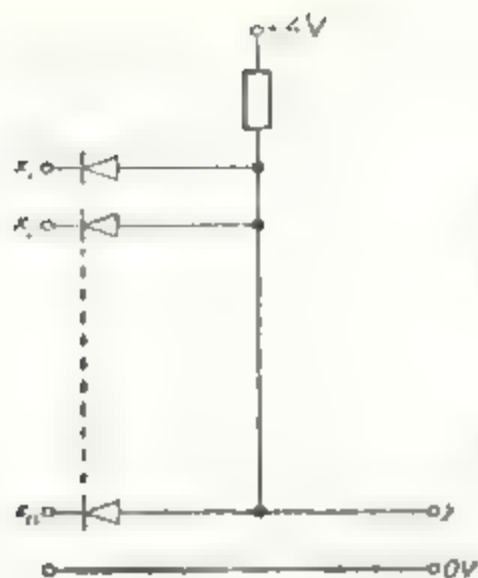


Fig. 1.50

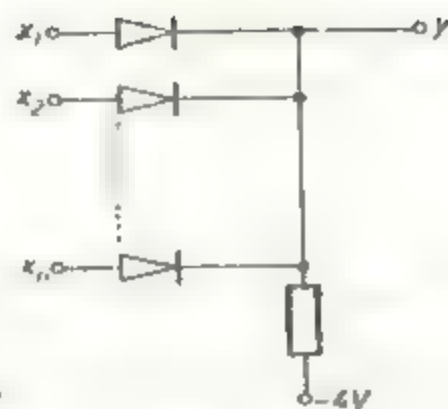


Fig. 1.51

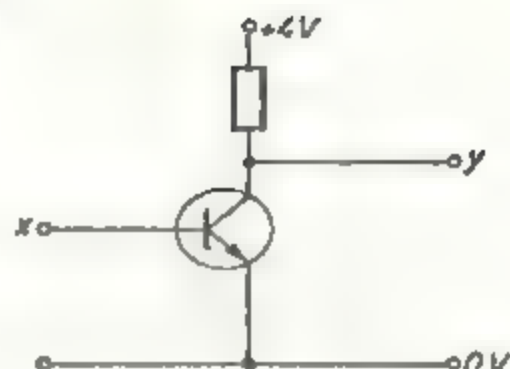


Fig. 1.52

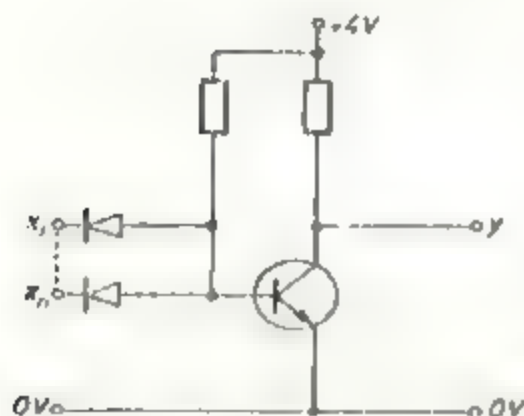


Fig. 1.53

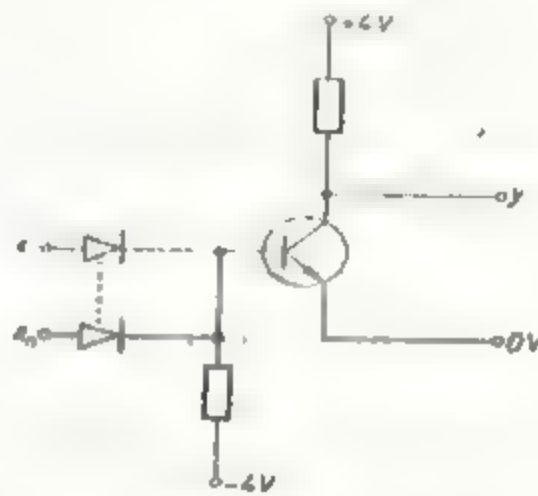


Fig. 1.54

Se pot realiza cu ușurință circuite logice universale cuplând un circuit NU cu un circuit SAU și respectiv cu un circuit ȘI. Se obțin astfel scheme pentru un circuit NU-ȘI (fig. 1.53) și pentru un circuit NICI (fig. 1.54).

Tehnica de calcul modernă folosește așa-numitele circuite integrate pentru realizarea funcțiilor booleene. Circuitele integrate sînt de o mare varietate. De obicei au un număr mare de borne (14, 16 sau 32). Unele circuite integrate realizează funcții simple, elementare. Însă, datorită numărului mare de borne, un circuit integrat poate conține mai multe circuite logice elementare independente. Există însă circuite integrate care realizează funcții complexe, cum ar fi codificatoare sau decodificatoare. De asemenea, există circuite integrate care au funcțiunile unor circuite secvențiale, avînd și posibilitatea de a memora valoarea unor semnale.

1.5.6. Exerciții

1. Să se găsească expresiile funcțiilor realizate de circuitele cu contacte din figura 1.55, să se simplifice și să se traseze schemele circuitelor care realizează aceleași funcții, conform expresiilor simplificate.
2. Să se afle expresiile funcțiilor realizate de circuitele din figura 1.56.

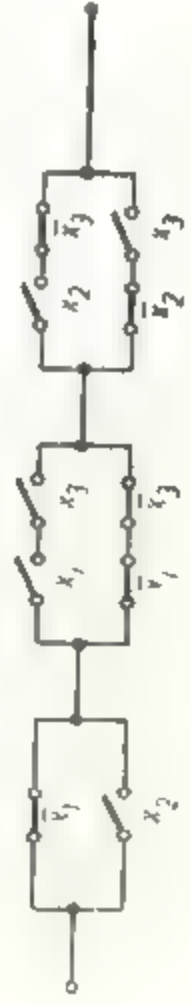
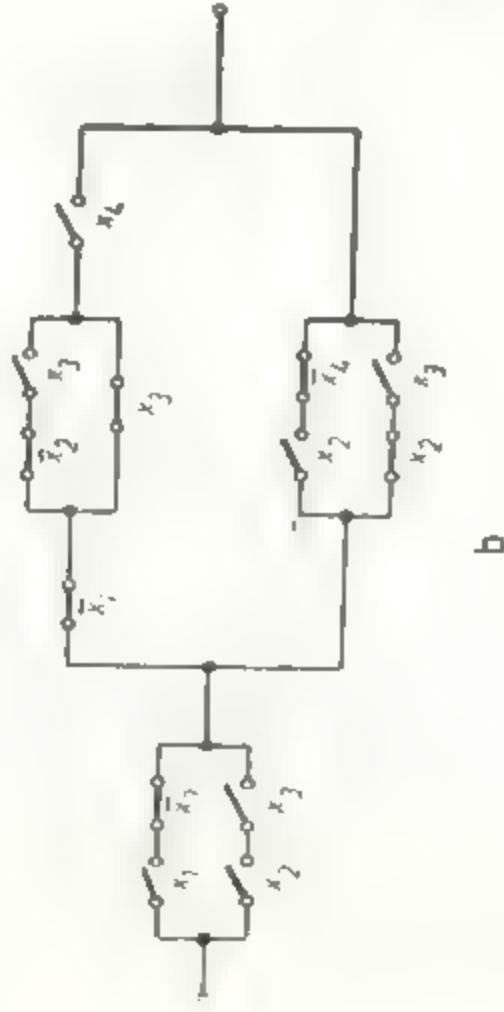
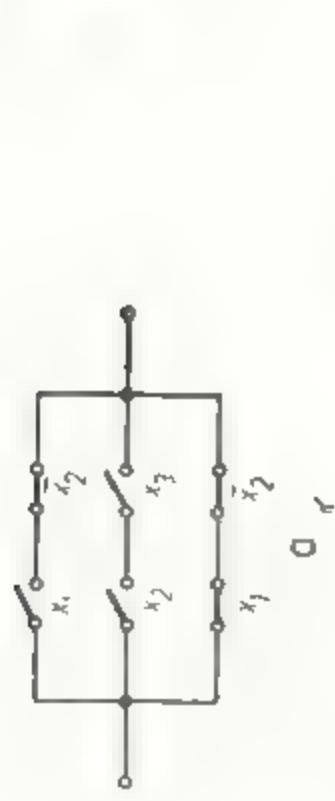


Fig. 1.55

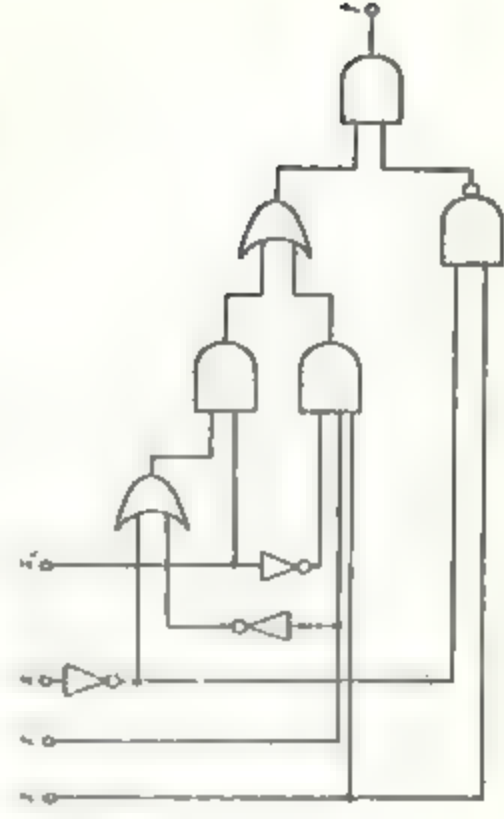
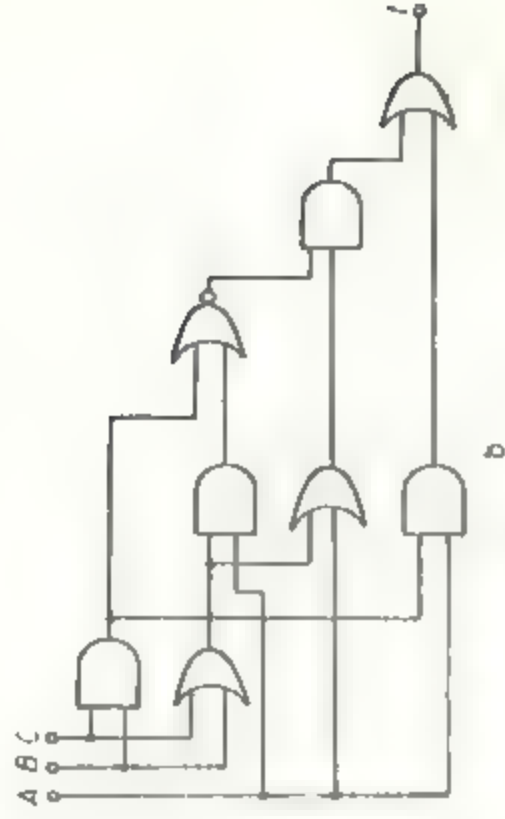
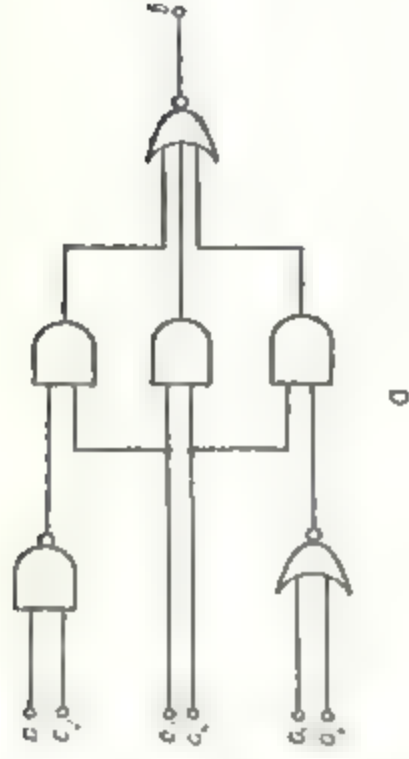


Fig. 1.56

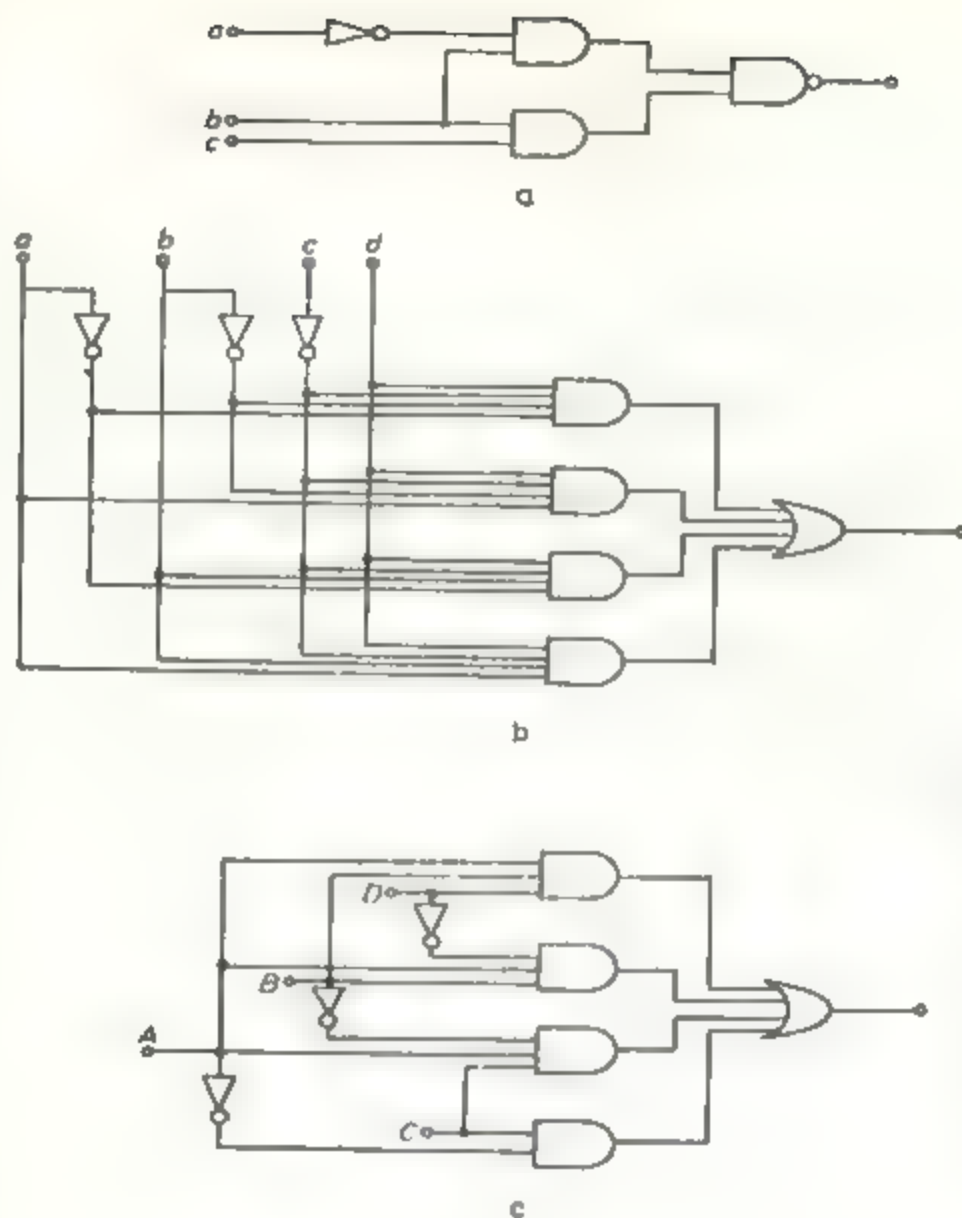


Fig. 1.57

cifra	x_1	x_2	x_3	x_4
0	0	0	1	1
1	0	1	0	0
2	0	1	0	1
3	0	1	1	0
4	0	1	1	1
5	1	0	0	0
6	1	0	0	1
7	1	0	1	0
8	1	0	1	1
9	1	1	0	0

Tabelul 1.19

- Să se simplifice funcțiile realizate de circuitele din figura 1.57 și să se traseze schema circuitelor care realizează aceleași funcții, conform expresilor simplificate.
- Corespunzător codului prezentat în tabelul 1.19, să se deseneze schema unui circuit:
 - decodificator;
 - codificator.
- Să se proiecteze un circuit logic care pentru trei semnale de intrare să indice prezența unui număr impar dintre acestea.
- Să se proiecteze un circuit logic cu trei borne de intrare, care să producă un semnal de ieșire egal cu valoarea majorității semnalelor de intrare.
- Să se conceapă un circuit logic cu patru borne de intrare. Două borne de intrare D_1 , D_2 sînt considerate borne de date, iar celelalte două C_1 , C_2 sînt considerate borne de control. Dacă ambele intrări de control au valoarea 0, ieșirea trebuie să aibă o valoare egală cu produsul logic al celor două intrări de date. Dacă ambele intrări de control au valoarea 1, atunci valoarea semnalului de ieșire trebuie să fie egală cu suma logică a semnalelor celor două intrări de date. Dacă numai o intrare de control are valoarea 1, atunci valoarea semnalului de la ieșire trebuie să fie egală cu valoarea semnalului de la intrarea de date corespunzătoare (D_1 pentru C_1 și D_2 pentru C_2).

II. GRAFURI NEORIENTATE

II.1. Noțiuni de bază

Un graf neorientat G este o pereche ordonată de mulțimi (X, U) , unde X este o mulțime finită, iar U este formată din perechi neordonate de elemente, din X . Putem considera deci că U este o familie de submulțimi cu două elemente din mulțimea X . Vom nota $G = (X, U)$.

Mulțimea X se numește *mulțimea vîrfurilor* sau a *nodurilor* grafului G și mulțimea U se numește *mulțimea muchilor* grafului G . O muchie fiind un element din U , ea este o submulțime cu două elemente din X , deci are forma $\{x, y\}$, unde $x, y \in X$.

Vom nota muchia $\{x, y\}$ prin $[x, y]$ și vom spune ca ea unește vîrfurile x și y . Deci notațiile $[x, y]$ și $[y, x]$ reprezintă aceeași muchie; vîrfurile x și y se numesc *extremitățile* acestei muchii.

Dacă $[x, y] \in U$ vom spune că vîrfurile x și y sînt *adiacente* în graful G , iar vîrfurile x și y sînt *incidente* cu muchia $[x, y]$.

Deci un graf G poate fi considerat ca o mulțime de vîrfuri, dintre care unele sînt unite două cîte două prin muchii.

Un graf G poate fi desenat în plan reprezentînd vîrfurile sale prin puncte și muchiile prin linii care unesc anumite perechi de vîrfuri.

Astfel graful $G = (X, U)$, unde $X = \{1, 2, \dots, 14\}$ și $U = \{[1, 2], [1, 3], [2, 4], [3, 4], [4, 5], [5, 6], [5, 7], [5, 8], [9, 14], [10, 14], [11, 14], [12, 14], [13, 14]\}$, se reprezintă ca în figura II.1.

De exemplu vîrfurile 4 este adiacent cu vîrfurile 2, 3 și 5, vîrfurile 8 este adiacent cu vîrfurile 5 iar vîrfurile 14 este adiacent cu vîrfurile 9, 10, 11, 12 și 13.

Gradul unui vîrf x este egal, prin definiție, cu numărul muchiilor incidente cu vîrfurile x și se notează cu $d(x)$.

De exemplu, pentru graful din figura II.1 obținem.

$$d(1) = d(2) = d(3) = 2; d(4) = 3; d(5) = 4; d(6) = d(7) = d(8) = 1.$$

Un vîrf cu gradul egal cu 1 se numește *vîrf terminal* al grafului. Pentru graful din figura II.1, vîrfurile 6, 7, 8, 9, 10, 11, 12, 13 sînt vîrfuri terminale. Un

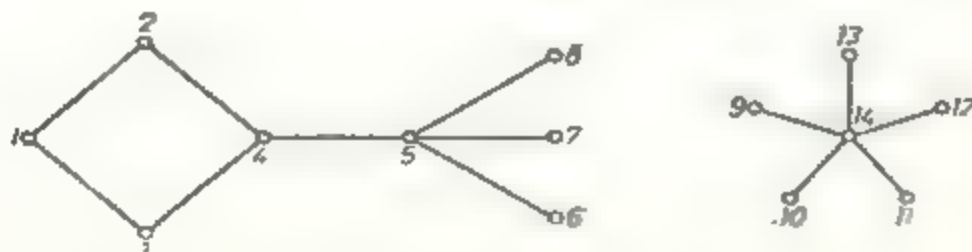


Fig. II.1

vîrf care are gradul egal cu zero, deci care nu mai este adiacent cu nici un alt vîrf al grafului, se numește *vîrf izolat*. Graful din figura 11.1 nu conține vîrfuri izolate.

Există o relație simplă între suma gradelor vîrfurilor unui graf și numărul de muchii, dată de propoziția următoare :

Propoziție. *Dacă graful G are m muchii și vîrfurile x_1, \dots, x_n , există relația :*

$$\sum_{i=1}^n d(x_i) = 2m.$$

Demonstrație Fiecare muchie $[x, y]$ a grafului G are doua extremități x și y , ea contribuind cu o unitate și la $d(x)$ și la $d(y)$.

Deci suma gradelor grafului este egală cu dublul numărului de muchii.

În particular, suma gradelor este un număr par. De aici mai rezulta următorul corolar.

Corolar. *Pentru orice graf G numărul vîrfurilor de grad impar este par*

Demonstrație. Să notăm cu S_1 suma gradelor pare și cu S_2 suma gradelor impare ale vîrfurilor grafului G .

Conform propoziției demonstrate putem scrie $S_1 + S_2 = 2m$. Să presupunem, prin reducere la absurd, ca numărul vîrfurilor de grad impar ale lui G este un număr impar. În acest caz S_2 este un număr impar. Însă S_1 este un număr par, deoarece fiecare termen din suma care îl definește pe S_1 este număr par. Am ajuns astfel la o contradicție și anume suma dintre un număr par, S_1 și un număr impar, S_2 , este un număr par, egal cu $2m$.

Rezulta ca presupunerea făcută nu este adevărată, deci proprietatea este demonstrată.

Pentru graful din figura 11.1 există 10 vîrfuri de grad impar și anume : 4, 6, 7, 8, 9, 10, 11, 12, 13, 14.

Un *graf parțial* al unui graf $G = (X, U)$, este un graf $G_1 = (X, V)$ care are aceeași mulțime de vîrfuri cu G , iar $V \subset U$. Deci, un graf parțial al lui G este G însuși sau se obține din G prin suprimarea anumitor muchii ale lui G .

Un *subgraf* al unui graf $G = (X, U)$ este prin definiție un graf $H = (Y, V)$, unde $Y \subset X$ iar muchiile din mulțimea V sînt toate muchiile din U care au ambele extremități în mulțimea de vîrfuri Y .

Deci un subgraf H al unui graf G este graful G însuși sau se obține din G prin suprimarea anumitor vîrfuri și a tuturor muchiilor incidente cu acestea.

Vom spune ca subgraful H este *indus* sau *generat* de mulțimea de vîrfuri Y .

Astfel, subgraful grafului G din figura 11.1, indus de mulțimea de vîrfuri $Y = \{1, 2, 3, 4, 5, 7\}$ este desenat în figura 11.2.

Un graf parțial al grafului din figura 11.2, obținut din acesta prin suprimarea muchiilor $[1, 3]$, $[3, 4]$ și $[4, 5]$ este desenat în figura 11.3.

Spunem că graful din figura 11.3 este un subgraf parțial al grafului din figura 11.1

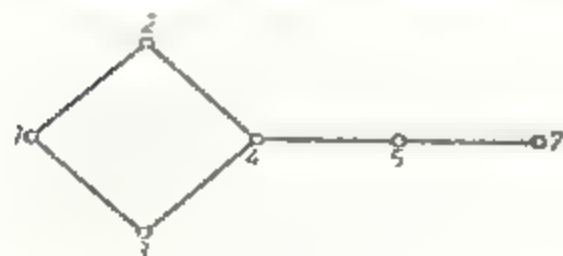


Fig. 11.2



Fig. 11.3

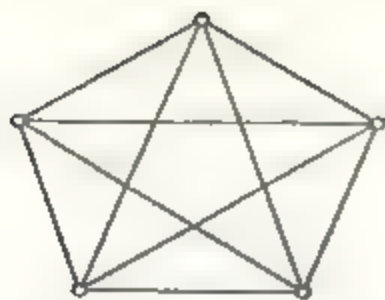


Fig. II.4

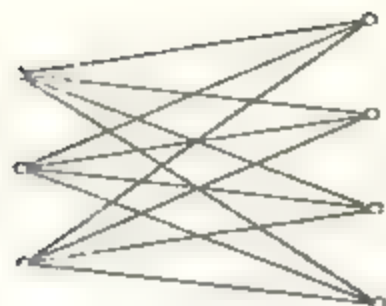


Fig. II.5

Un graf cu n vîrfuri pentru care oricare două vîrfuri sînt adiacente se numește *graf complet* cu n vîrfuri și se notează prin K_n .

În figura II.4 este reprezentat graful K_5 .

Deoarece pentru graful K_n oricare două vîrfuri sînt adiacente, rezultă că numărul m de muchii ale acestui graf este egal cu numărul submulțimilor cu 2 elemente ale unei mulțimi cu n elemente, adică $m = C_n^2$.

Un graf G se numește *bipartit* dacă există o partiție a mulțimii vîrfurilor :

$$X = X_1 \cup X_2, X_1 \cap X_2 = \emptyset$$

astfel încît fiecare muchie a grafului unește un vîrf din X_1 cu un vîrf din X_2 .

Dacă X_1 are p elemente, X_2 are q elemente și oricare vîrf din X_1 este adiacent cu toate vîrfurile din X_2 , graful se numește *bipartit complet* și se notează prin $K_{p,q}$. Graful $K_{3,3}$ este desenat în figura II.5

Deoarece fiecare vîrf $x \in X_1$ are gradul $d(x) = q$, rezultă că numărul de muchii ale grafului $K_{p,q}$ este egal cu pq .

Un *lanț* este un șir (succesiune) de vîrfuri :

$$L = [x_0, x_1, \dots, x_r]$$

cu proprietatea că oricare două vîrfuri vecine sînt adiacente, adică $[x_0, x_1], [x_1, x_2], \dots, [x_{r-1}, x_r] \in U$. Vîrfurile x_0 și x_r se numesc *extremitățile lanțului* L , iar numărul r se numește *lungimea* acestui lanț. Dacă vîrfurile x_0, x_1, \dots, x_r sînt distincte două cîte două, lanțul L se numește *elementar*.

Pentru graful din figura II.1 următoarele șiruri de vîrfuri sînt lanțuri :

$$L_1 = [1, 2, 4, 5, 6], L_2 = [4, 5, 8, 5, 6], L_3 = [9, 14, 10],$$

$$L_4 = [9, 14, 10, 14, 11], L_5 = [1, 2, 4, 3, 1].$$

Lanțurile L_1 și L_3 sînt lanțuri elementare, deoarece conțin numai vîrfuri distincte două cîte două. Un lanț $L = [x_0, \dots, x_r]$ poate fi interpretat ca traseul unei deplasări pe muchiile grafului în ordinea $[x_0, x_1], [x_1, x_2], \dots, [x_{r-1}, x_r]$. De aceea lanțul L de extremitați x_0 și x_r se mai spune că este un lanț de la x_0 la x_r sau de la x_r la x_0 . Lungimea lanțului L este deci numărul de parcurgeri ale muchiilor grafului G .

Dacă $x_0 = x_r$ și toate muchiile $[x_0, x_1], [x_1, x_2], \dots, [x_{r-1}, x_r]$ sînt distincte două cîte două, lanțul L se numește *ciclu*. Dacă toate vîrfurile ciclului, cu excepția primului și a ultimului vîrf, sînt distincte două cîte două, ciclul se numește *elementar*.

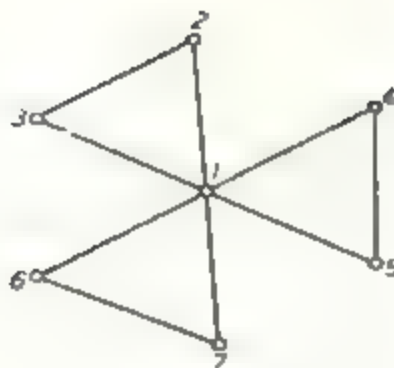


Fig. 11.6

Astfel, lanțul L , este un ciclu elementar care trece prin vîrfurile 1, 2, 4, 3.

Lanțurile $[4, 5, 4]$ sau $[1, 2, 4, 5, 4, 3, 1]$ nu sînt cicluri deoarece folosesc de mai multe ori aceeași muchie.

Pentru graful din figura 11.6 obținem trei cicluri elementare:

$$C_1 = [1, 2, 3, 1], C_2 = [1, 4, 5, 1] \text{ și } C_3 = [1, 6, 7, 1]$$

Deoarece nu contează sensul de deplasare pe fiecare muchie, aceste cicluri pot fi scrise și sub forma:

$$C_1 = [1, 3, 2, 1], C_2 = [1, 5, 4, 1] \text{ și } C_3 = [1, 7, 6, 1],$$

sau alegînd alte vîrfuri ca primele vîrfuri în scrierea ciclului. De exemplu:

$$C_1 = [2, 1, 3, 2] \text{ sau } C_1 = [2, 3, 1, 2].$$

Ciclul $C_1 = [1, 2, 3, 1, 4, 5, 1]$ nu este elementar, deoarece vîrfurile 1, care este primul și ultimul vîrf, se mai repetă o dată în acest șir.

Un graf G se numește *conex* dacă pentru orice pereche de vîrfuri $\{x, y\}$ cu $x \neq y$, există un lanț de la x la y .

Graful G din figura 11.1 nu este conex, deoarece nu există nici un lanț între un vîrf din mulțimea $X_1 = \{1, 2, 3, 4, 5, 6, 7, 8\}$ și un vîrf din mulțimea $X_2 = \{9, 10, 11, 12, 13, 14\}$. Fiecare din mulțimile X_1 și X_2 induce un subgraf conex al grafului G .

Aceste două subgrafuri conexe se numesc *componentele conexe* ale grafului G .

În general, o componentă conexă C a unui graf G se definește ca fiind un subgraf conex al lui G care este maximal în raport cu această proprietate, adică nu există nici un lanț al lui G care să unească un vîrf din C cu un vîrf care nu aparține lui C .

Pentru graful G din figura 11.1 mulțimea de vîrfuri $\{6, 7, 8\}$ nu induce o componentă conexă deoarece nu induce un subgraf conex. Mulțimea de vîrfuri $Y = \{9, 10, 14\}$ care induce un subgraf conex nu formează o componentă, deoarece nu este maximală în raport cu această proprietate. Într-adevăr, există de exemplu muchia $[14, 13]$ care unește vîrfurile 14 din Y cu vîrfurile 13 care nu aparține lui Y .

Graful din figura 11.2 este conex, iar graful din figura 11.3 are trei componente conexe: una este formată din vîrfurile izolate 3, alta este formată din vîrfurile 1, 2, 4 și cea de a III-a are vîrfurile 5, 7.

Exemple

1. În figura 11.7 este reprezentată o parte a schemei căilor ferate din țara noastră.

Acest desen este un graf, vîrfurile sale reprezentînd nodurile de cale ferată, iar muchiile reprezentînd legăturile directe pe calea ferată dintre două noduri. Dacă cunoaștem distanțele în kilometri asociate fiecărei muchii, ne putem pune problema găsirii celui mai scurt traseu pe calea ferată între două localități.

Acesta va corespunde unui lanț elementar în graful din figura 11.7, care unește cele două localități și pentru care suma distanțelor asociate muchiilor este minimă.

O excursie în circuit care trece o singură dată prin anumite localități, întorcîndu-se în localitatea de pornire, va corespunde unui ciclu elementar în acest graf.

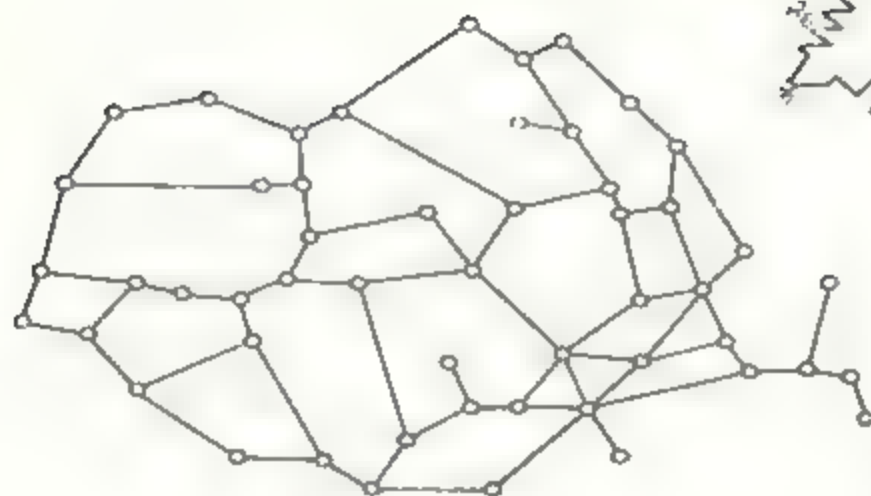


Fig. II.7

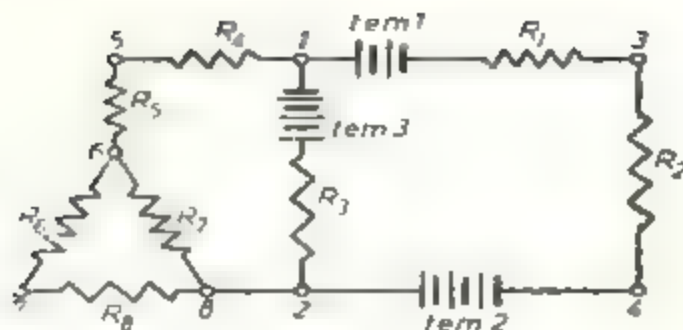


Fig. II.8



Fig. II.9

2. Vom considera acum un exemplu din fizică și anume calculul intensităților curenților care trec prin ramurile unei rețele electrice, ca cea din figura II.8

Pentru a rezolva această problemă, cunoscând schema rețelei, tensiunile electromotoare și valorile rezistențelor, se scriu legile lui Kirchhoff relative la noduri și la ochiuri de rețea.

Făcând abstracție de elementele de circuit care se găsesc pe laturile schemei putem desena această rețea sub forma grafului din figura II.9. Nodurile rețelei vor corespunde vîrfurilor grafului din figura II.9, iar ochiurile de rețea vor corespunde ciclurilor elementare ale acestui graf.

Fizicianul Kirchhoff a studiat, la mijlocul secolului trecut, rețelele electrice ca metode care aparțin astăzi teoriei grafurilor, contribuind la dezvoltarea acestei teorii.

3. Formulele de structură ale substanțelor chimice sînt grafuri pentru care legăturile dintre vîrfuri corespund legăturilor dintre grupările sau atomii care compun molecula.

Astfel, apa are molecula reprezentată în figura II.10 a, acetilena are formula de structură în figura II.10 b, molecula de benzen este reprezentată în figura II.10 c, iar cea de glucoză în figura II.10.d.

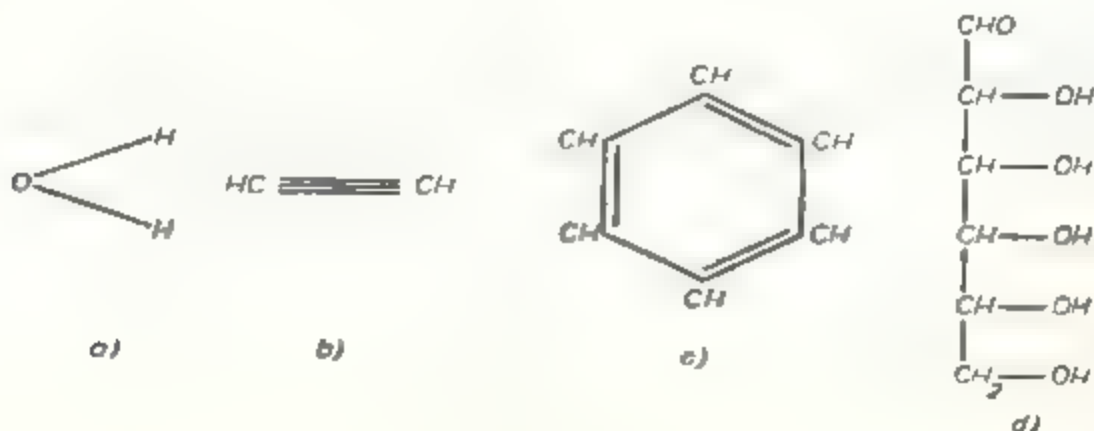


Fig. II.10

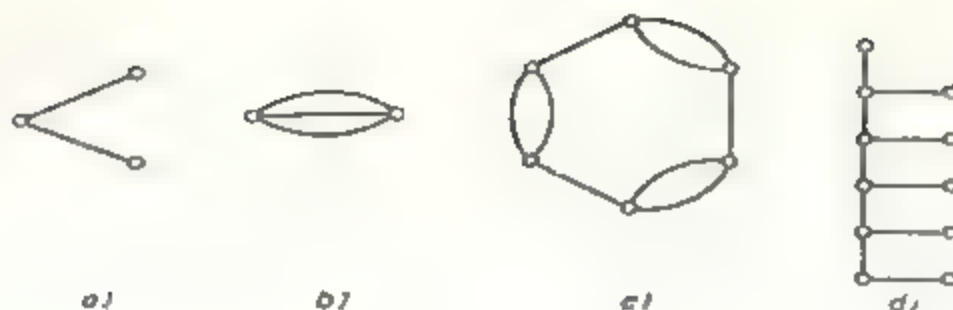


Fig. II 11

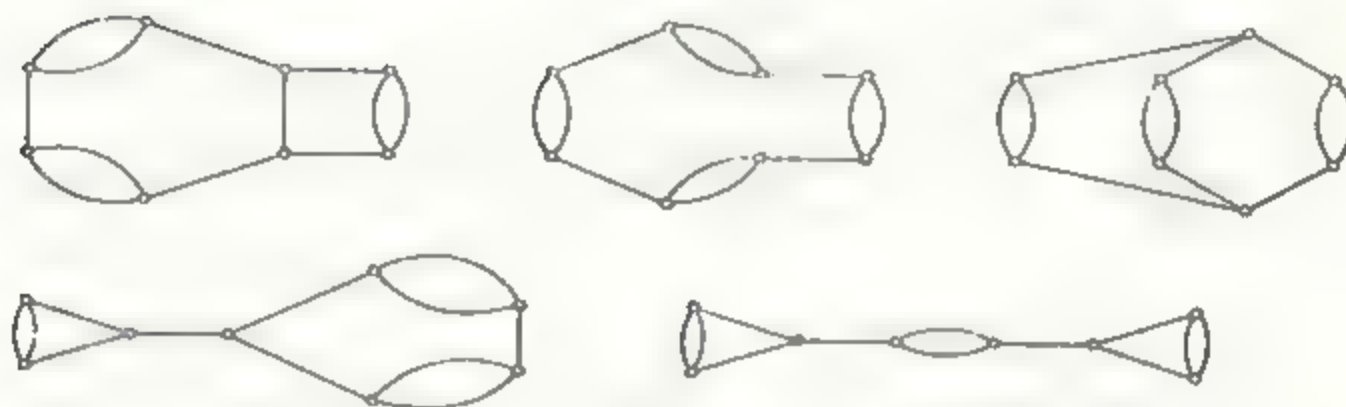


Fig. II 12

În figurile II 11 a – II 11 d aceste formule de structură au fost reprezentate sub formă de grafuri pentru care vîrfurile sînt atomii (respectiv grupările) din moleculă, muchiile reprezentînd legăturile lor chimice.

Grafurile din figurile II.11.b și II.11.c nu sînt grafuri în sensul definiției date, deoarece între anumite perechi de vîrfuri există mai multe muchii.

Un astfel de graf cu muchii multiple se numește *multigraf*.

Într-un graf sau multigraf care este desenul moleculei unei substanțe, gradul unui vîrf este tocmai valența atomului (grupării) respective.

În figura II 12 sînt reprezentați cinci izomeri ai compusului organic numit ciclooctatetrena, care are formula C_8H_8 . Fiecare din cele cinci multigrafuri conține cîte trei muchii duble și șase muchii simple, în vîrfurile acestora găsindu-se gruparea CH , de valență egală cu trei. Din acest motiv fiecare vîrf este incident cu exact trei muchii, deci are gradul trei.

Acești izomeri pot trece unii în alții prin acțiunea diferiților factori exteriori și enumerarea tuturor izomerilor posibili, deci a tuturor multigrafurilor cu un număr fixat de vîrfuri, toate avînd gradul trei, poate da sugestii despre obținerea lor în laborator.

Probleme

- II 1 1. Să se determine lanțurile elementare dintre vîrfurile 1 și 7 ale grafului din figura II 13.
 II.1 2. Dacă un graf G nu conține cicluri, orice lanț care nu folosește de mai multe ori o aceeași muchie este elementar.
 II 1.3. Un graf G cu n vîrfuri are m muchii astfel încît să aibă loc inegalitatea:

$$m \geq C_{n-1}^2$$

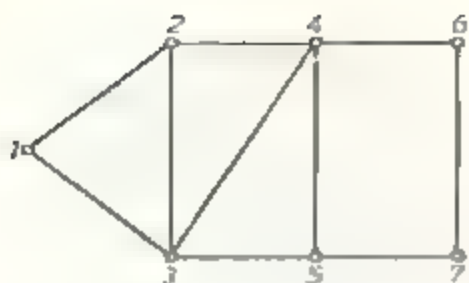


Fig. II.13



Fig. II.14

Să se arate că G nu are vîrfuri izolate.

- II 1 4. Un graf G are n vîrfuri și p componente conexe. Să se arate că numărul m al muchiilor grafului G verifică inegalitatea:

$$m \leq C_{n-1}^2, \dots$$

Să se indice pentru ce grafuri această inegalitate devine egalitate.

- II 1 5. Să se arate că orice graf conține cel puțin două vîrfuri care au același grad.

- II 1 6. Două grafuri $G = (X, U)$ și $H = (Y, V)$ se numesc izomorfe dacă există o bijecție:

$$f: X \rightarrow Y$$

astfel încît $[x, y] \in U$ dacă și numai dacă $[f(x), f(y)] \in V$. Deci două grafuri izomorfe au același număr de vîrfuri și se obțin unul din celălalt printr-o renumerotare a vîrfurilor. Să se arate că grafulurile din figura II 14 sînt izomorfe. Aceeași problemă pentru grafulurile din figura II.15.

- II 1 7. Fînd dat un graf $G = (X, U)$, complementarul său $\bar{G} = (X, \bar{U})$ se definește ca fiind graful cu aceeași mulțime X de vîrfuri, două vîrfuri fiind adiacente în \bar{G} dacă și numai dacă ele nu sînt adiacente în G .

De exemplu unul din cele două grafuri din figura II 14 este complementarul celui alt.

Să se arate că dacă G nu este conex, atunci complementarul său \bar{G} este conex.

- II 1 8. Să se calculeze numărul grafulor cu n vîrfuri date x_1, x_2, \dots, x_n .

- II 1 9. Pe mulțimea X a vîrfurilor unui graf $G = (X, U)$ se introduce următoarea relație binară:

Spunem că x este în relație cu y și scriem $x \sim y$ dacă $x = y$ sau există un lanț de extremități x și y . Să se arate că această relație binară este o relație de echivalență și clasele acestei echivalențe sînt mulțimile de vîrfuri ale componentelor conexe ale grafului.

- II 1 10. Fie G un graf care nu conține călări elementare de lungime pară. Să se arate că G conține un vîrf x de grad $d(x) \leq 2$.

- II 1 11. Să se găsească toate subgrafurile complete cu 3 vîrfuri ale grafului din figura II.16.

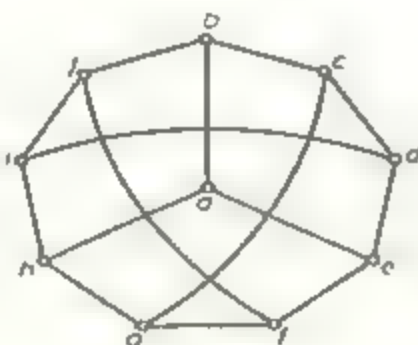
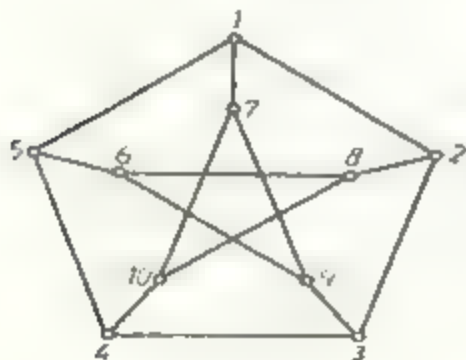


Fig. II.15

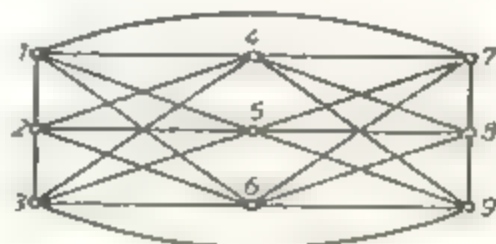


Fig. II.16

II.2. Grafuri hamiltoniene

Un ciclu elementar al unui graf G care trece prin toate vîrfurile grafului se numește *ciclu hamiltonian*.

Un graf G care are un ciclu hamiltonian se numește *graf hamiltonian*.

Originea acestui termen se găsește într-un joc inventat în anul 1857 de matematicianul William Hamilton. Partea sa principală era un dodecaedru regulat făcut din lemn (fig. II.17).

Acesta este un poliedru cu 12 fețe care sînt toate pentagoane regulate, iar în fiecare din cele 20 de vîrfuri se întîlnesc cite 3 muchii. Fiecare vîrf al dodecaedrului lui Hamilton era marcat cu numele unui oraș. Jocul consta în găsirea unui drum de-a lungul muchilor dodecaedrului care să treacă prin fiecare din cele 20 de orașe exact o dată și să se întoarcă în orașul din care a plecat.

Pentru a ușura memorarea trecerilor efectuate, în fiecare vîrf al dodecaedrului era cîte un cui cu o sfoară mare, astfel încît în jurul acestor cuie putea să se întindă un fir care să indice drumul parcurs în această călătorie imaginară în jurul lumii.

Problema revine la găsirea unui ciclu hamiltonian în graful format cu vîrfurile și muchiile dodecaedrului. Acest graf este hamiltonian, un ciclu hamiltonian în reprezentarea plană a grafului dodecaedrului fiind desenat cu linii îngroșate în figura II.18.

O problemă mai generală este aceea a voiajorului comercial, care are următorul enunț: Un voiajor comercial trebuie să prezinte în n orașe produsele fabricii pe care o reprezintă, după care se întoarce în orașul din care a plecat. Cunoscîndu-se costul deplasării între oricare doua dintre cele n orașe, se cere să se determine un traseu care să viziteze o singură dată cele n orașe și care să aibă un cost total minim.

În termenii teoriei grafurilor, problema revine la determinarea unui ciclu hamiltonian în graful complet K_n ale cărui vîrfuri reprezintă cele n orașe, pentru care suma costurilor asociate celor n muchii ale ciclului să fie minimă.

Pentru a rezolva practic această problemă ar trebui definit un algoritm eficient de rezolvare.

Deși această problemă a fost mult studiată, nu a fost descoperit pînă în prezent un algoritm eficient de rezolvare și nici nu se cunoaște dacă poate exista un astfel de algoritm pentru găsirea unui ciclu hamiltonian de cost minim.

Ciclurile hamiltoniene în grafuri particulare au fost de fapt studiate cu mult înainte ca Hamilton să fi propus jocul său.

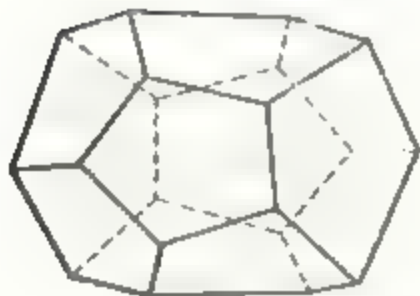


Fig. II.17

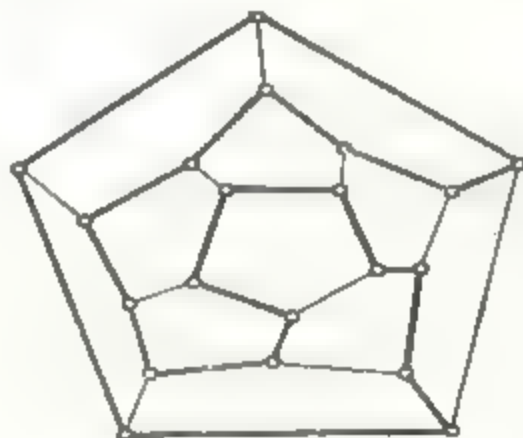


Fig. II.18

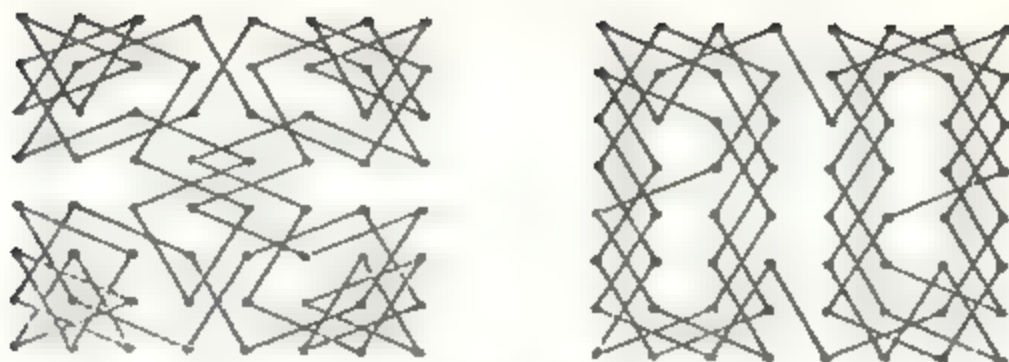


Fig. 11.19

În special jocul calului pe o tabla de șah a fost analizat de Euler în anul 1759. Acest joc cere să se găsească un ciclu hamiltonian în graful cu 64 de vârfuri care reprezintă cele 64 de pătrate ale unei table de șah și pentru care două vârfuri sînt adiacente dacă un cal de pe tabla de șah poate sari în L de pe un pătrat pe celalalt. În figura 11.19 sînt reprezentate două soluții ale acestui joc.

Teorema următoare ne da o condiție suficientă pentru existența unui ciclu hamiltonian.

Teoremă. Dacă G este un graf cu $n \geq 3$ vârfuri astfel încît gradul oricărui vîrf x verifică inegalitatea :

$$d(x) \geq \frac{n}{2},$$

rezultă că G este hamiltonian.

Demonstrație. Vom raționa prin reducere la absurd. Să presupunem deci că pentru orice vîrf x avem $d(x) \geq \frac{n}{2}$ și G nu conține nici un ciclu hamiltonian.

Vom adăuga muchii între perechi de vîrfuri neadiacente atît timp cît aceasta este posibil, fără ca în graful astfel obținut să existe un ciclu hamiltonian. Se obține astfel un graf H cu proprietatea că $d(x) \geq \frac{n}{2}$ pentru orice vîrf x , deoa-

rece prin adăugarea de muchii gradele vîrfurilor cresc. În plus, pentru orice pereche de vîrfuri neadiacente x și y , prin adăugarea muchiei $[x, y]$ se creează un ciclu hamiltonian care folosește muchia $[x, y]$. Deci în graful H există un lanț elementar de extremități x și y :

$$L = [x_1, x_2, \dots, x_n],$$

unde $x_1 = x$, $x_n = y$ și vîrfurile x_1, \dots, x_n sînt toate cele n vîrfuri ale grafului H .

Este clar că există macar o pereche de vîrfuri neadiacente x, y în graful H , deoarece în caz contrar $H = K_n$ și graful complet cu n vîrfuri este hamiltonian.

Să notăm $d(x) = k$ și fie $x_{i_1}, x_{i_2}, \dots, x_{i_k}$ unde $i_1 = 2 < i_2 < \dots < i_k$, vîrfurile adiacente cu vîrfurile x .

Deoarece H nu este hamiltonian, rezulta că y nu este adiacent cu nici unul din vîrfurile $x_{i_1-1}, \dots, x_{i_k-1}$. Într adevăr, în caz contrar rezultă că x este adiacent cu un vîrf x_{i_1} și y este adiacent cu x_{i_1-1} , ceea ce ar produce un ciclu hamiltonian în H . Acest ciclu este desenat cu linie îngroșată în figura 11.20.

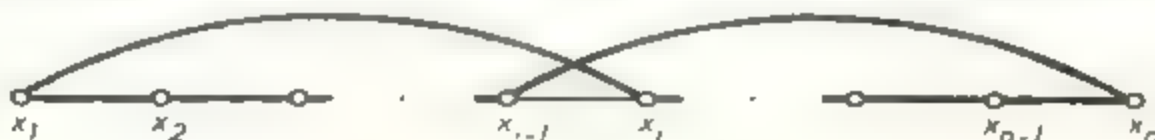


Fig. 11.20

Însă această proprietate contrazice ipoteza făcută că H nu este hamiltonian. Deci dintre cele $n - 1$ vîrfuri: x_1, \dots, x_{n-1} , vîrfurile $x_n = y$ nu este adiacent cu cel puțin k vîrfuri, adică:

$$d(y) \leq n - 1 - k \leq n - 1 - \frac{n}{2} = \frac{n}{2} - 1,$$

deoarece în graful H avem $d(x) = k \geq \frac{n}{2}$.

Am obținut deci $d(y) < \frac{n}{2}$, ceea ce contrazice proprietatea $d(y) \geq \frac{n}{2}$, valabilă pentru orice vîrf din H . Deci am demonstrat prin reducere la absurd că H este hamiltonian.

Probleme

- 11.2.1. Care din cele două grafuri din figura 11.15 au un ciclu hamiltonian?
Dar graful din figura 11.21?

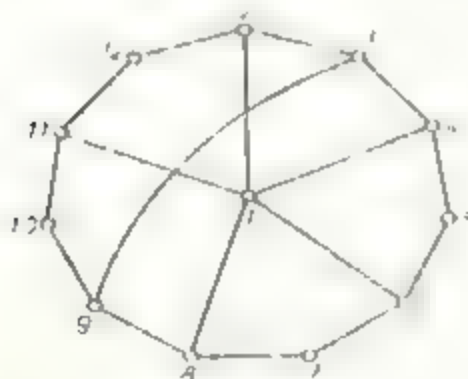


Fig. 11.21

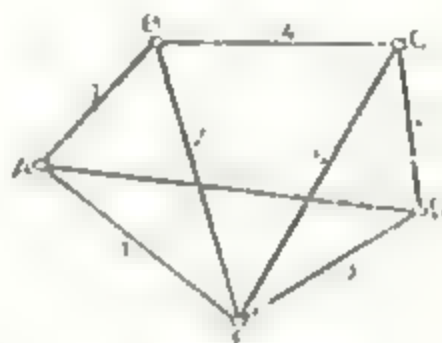


Fig. 11.22

- 11.2.2. Un delegat al unei întreprinderi din orașul A trebuie să meargă în orașele B, C, D și E și apoi să se întoarcă în orașul A .

Cunoscînd costurile deplasării dintr-un oraș în altul, care sînt numerele asociate muchiilor grafului din figura 11.22, să se determine succesiunea de orașe pentru care costul total

al deplasării să fie minim, știind că sosirea în orașul B este mai urgentă decît sosirea în orașul D .

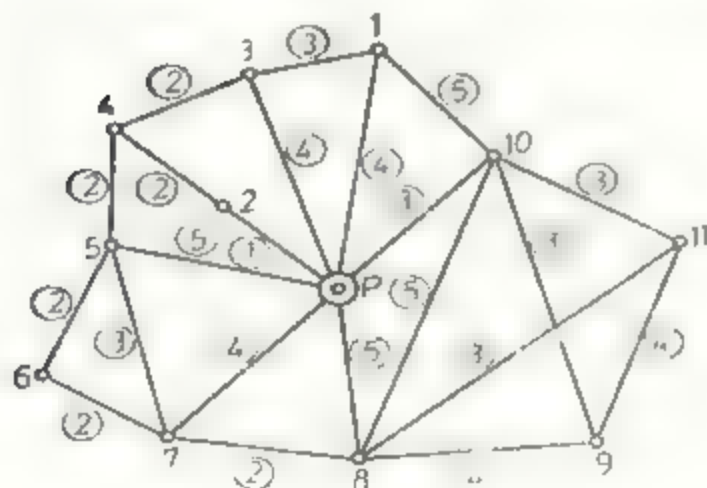


Fig. 11.23

- 11.2.3. O mașină poștală trebuie să plece de la oficiul poștal P , să vîdeze 11 cutii poștale și să se întoarcă la oficiul P . Cele 11 cutii sînt reprezentate ca vîrfuri ale grafului din figura 11.23, iar distanțele dintre cutii sînt reprezentate prin numere în cerceule asociate muchiilor acestui graf, care indică posibilitățile de deplasare de la o cutie la alta. Să se găsească un ciclu hamiltonian în acest graf, care să corespundă unui traseu al mașinii poștale de lungime totală minimă.

11.2.4. Să se arate că numărul ciclurilor hamiltoniene ale grafului complet K_n cu $n \geq 3$ vârfuri este egal cu

$$\frac{(n-1)!}{2}.$$

11.2.5. Să se arate că numărul ciclurilor elementare ale grafului complet K_n este egal cu

$$\sum_{k=3}^n \frac{n(n-1)\dots(n-k+1)}{k}$$

pentru orice $n \geq 3$.

11.3. Grafuri euleriene

Un ciclu al unui graf G care conține toate muchiile lui G se numește *ciclu eulerian*. Un graf G care are un ciclu eulerian se numește *graf eulerian*.

Din definiția unui ciclu toate muchiile pe care acesta le conține sînt distincte două cîte două. Deci putem spune că un ciclu hamiltonian trece o singură dată prin toate vîrfurile unui graf, în timp ce un ciclu eulerian trece o singură dată prin toate muchiile unui graf.

Ciclurile euleriene își trag denumirea de la numele matematicianului Leonard Euler care în anul 1736 a caracterizat grafurile care au un astfel de ciclu. Euler a fost condus la această problemă de jocul celor 7 poduri din orașul Kaliningrad. Cele 7 poduri sînt desenate în figura 11.24 și problema era următoarea:

Se poate realiza o plimbare peste toate cele 7 poduri, trecînd o singură dată peste fiecare pod?

Am reprezentat cele 4 regiuni A, B, C, D și cele 7 poduri a, b, c, d, e, f, g ca vîrfuri ale grafului din figura 11.25, muchiile grafului reprezentînd posibilitățile de trecere de pe un mal pe un pod sau reciproc.

Problema celor 7 poduri are o soluție dacă există un ciclu eulerian pentru graful din figura 11.25. Un astfel de ciclu la fiecare trecere printr-un vîrf utilizează două muchii, care nu mai pot fi folosite pentru o nouă trecere. Existența unui ciclu eulerian pentru acest graf este imposibilă, deoarece, de exemplu, gradul vîrfului D este egal cu 3.

La o primă trecere prin D sînt utilizate două din cele trei muchii, o nouă trecere fiind deci imposibilă și una din muchiile incidente cu D rămîne nefolosită. Această observație simplă ne conduce la următoarea caracterizare a grafurilor euleriene:

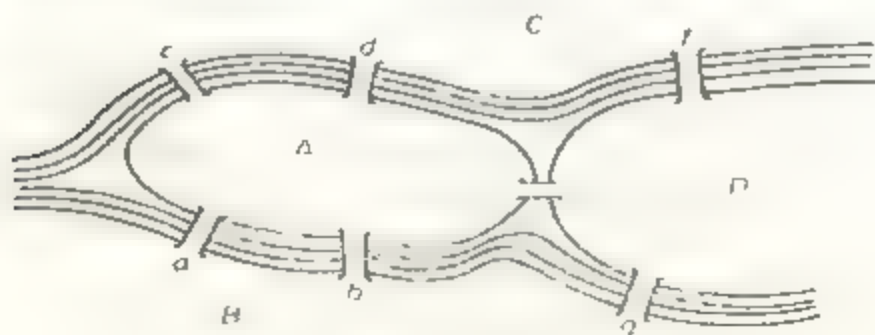


Fig. 11.24

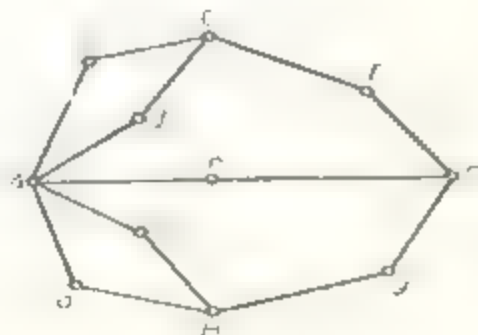


Fig. 11.25

Teoremă. Un graf G fără vîrfuri izolate este eulerian dacă și numai dacă este conex și gradele tuturor vîrfurilor sale sînt numere pare.

Demonstrație. Să presupunem ca graful G nu are vîrfuri izolate și conține un ciclu eulerian. Fie x, y două vîrfuri oarecare ale lui G . Dacă x și y sînt adiacente, rezultă că există un lanț de lungime egală cu unu între x și y . În caz contrar, deoarece G nu are vîrfuri izolate, există două muchii u și v astfel încît u este incidentă cu x și v este incidentă cu y . G fiind eulerian, există un ciclu care trece prin u și v , deci prin vîrfurile x și y . Rezultă că x și y sînt unite printr-un lanț, deci G este conex.

Dacă ciclul eulerian este $[x_1, x_2, \dots, x_m, x_1]$ și dacă vîrful x apare de k ori în șirul x_1, x_2, \dots, x_m , rezultă $d(x) = 2k$, deoarece fiecare trecere printr-un vîrf utilizează două muchii. Deci toate gradele vîrfurilor lui G sînt numere pare.

Pentru a demonstra suficiența condiției, sa presupunem că graful G nu are vîrfuri izolate, este conex și are gradele tuturor vîrfurilor numere pare. Să arătăm ca G conține un ciclu eulerian. Să presupunem, prin reducere la absurd, ca G nu conține un ciclu eulerian. Deci sau G nu conține cicluri, sau conține numai cicluri care nu parcurg toate muchiile lui G . Fie C un ciclu al grafului G care conține un număr maxim de muchii ale lui G . Vom arăta că presupunerea că C nu există sau C nu conține toate muchiile lui G ne conduce la o contradicție.

Pentru aceasta vom considera graful parțial H al lui G obținut din G prin suprimarea tuturor muchiilor parcurse de ciclul C . Deoarece ciclul C folosește în fiecare vîrf x al lui G un număr par de muchii și conform ipotezei gradele vîrfurilor lui G sînt numere pare, rezultă că gradele tuturor vîrfurilor lui H sînt numere pare, ca diferențe de numere pare. Ciclul C nefiind eulerian, rezultă că H are mulțimea muchiilor nevidă și măcar una din muchiile lui H are în comun una din extremitățile sale, fie z , cu ciclul C . Într-adevar, în caz contrar ar rezulta că mulțimea vîrfurilor parcurse de ciclul C ar forma o componentă conexa care nu conține toate vîrfurile lui G . Însă acest lucru contrazice ipoteza făcută ca G este conex. Sa plecăm din vîrfurile z pe muchia lui H incidentă cu z , deplasîndu-ne pe muchiile grafului H , fără a trece de două ori pe aceeași muchie. Dacă G nu conține cicluri, obținem $H = G$ și alegem pe z un vîrf oarecare al lui G . După un număr finit de astfel de deplasări ne vom întoarce în z . Pentru a justifica această afirmație, să observăm ca gradele grafului H fiind numere pare, iar fiecare trecere printr-un vîrf utilizînd exact două muchii care nu mai pot fi parcurse, rămîn tot un număr par de muchii. Deci odată ajunși în oricare vîrf diferit de z al lui H mai rămîn un număr impar, deci nenul, de muchii neutilizate și putem părăsi acel vîrf. Deoarece H are un număr finit de muchii,

rezultă că după un număr de pași ne reîntoarcem în z .

Am obținut astfel un ciclu C_1 în graful H , care nu are nici o muchie în comun cu ciclul C . Reuniunea muchiilor lui C și C_1 generează un nou ciclu în graful G care conține un număr mai mare de muchii ca C (fig. II.26). S-a ajuns astfel la o contradicție, deoarece s-a presupus că C conține un număr maxim de muchii. Rezultă că G este eulerian și teorema este demonstrată.

Ordinea de parcurgere a muchiilor lui C și C_1 pentru a obține un ciclu mai lung ca C este de exemplu următoarea:



Fig. II.26

Plecăm din z și ne întoarcem în z pe muchiile ciclului C , apoi facem aceeași operație pe muchiile lui C_1 . Procedeu constructiv utilizat pentru a demonstra suficiența teoremei constituie în același timp un algoritm pentru a obține cicluri din ce în ce mai lungi, până la obținerea unui ciclu eulerian într-un graf conex și cu gradele vîrfurilor pare. Ciclurile euleriene intervin într-o serie de probleme de colectare și distribuire. Condiția impusă unui ciclu de a nu parcurge de mai multe ori o aceeași muchie se poate traduce printr-o condiție de lungime minimă a traseului.

Astfel, un poștaș care pleacă de la un oficiu poștal pentru a distribui corespondența pe un număr de străzi și se întoarce la oficiu, trebuie să parcurgă un ciclu eulerian în graful care reprezintă rețeaua stradală respectivă, pentru a avea de străbătut un traseu de lungime minimă.

Probleme

11.3.1 Să se găsească un ciclu eulerian pentru graful din figura 11.27

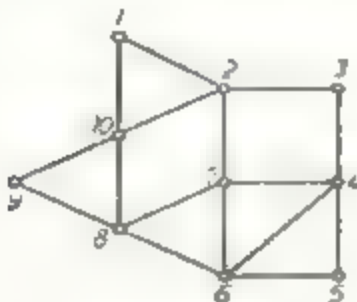


Fig. 11.27

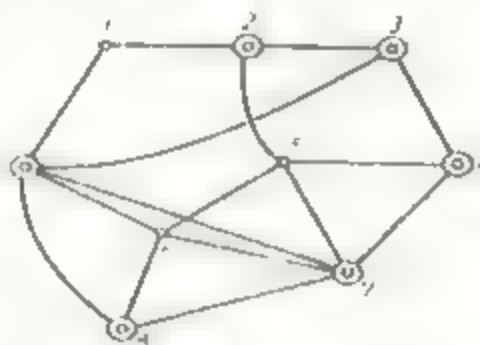


Fig. 11.28

11.3.2 Să se indice care este numărul minim de muchii care trebuie adăugate grafului conex din figura 11.28 pentru a-l transforma într-un graf eulerian.

11.3.3 Să se indice cum poate fi transformat graful neconex din figura 11.29 într-un graf eulerian prin adăugarea unui număr minim de muchii.

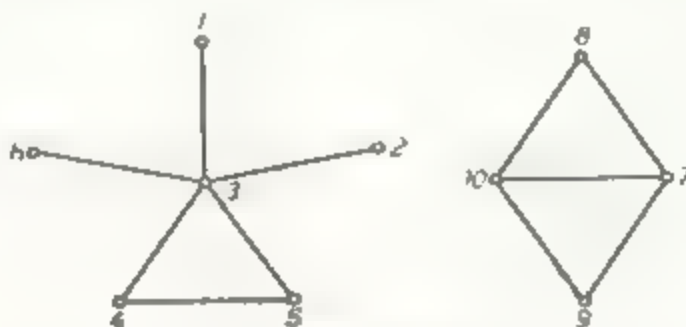


Fig. 11.29

11.3.4 Un graf G conține o mulțime de cicluri elementare astfel încât fiecare muchie a lui G aparține exact unuia din aceste cicluri elementare dacă și numai dacă toate gradele vîrfurilor lui G sînt numere pare.

11.3.5 Fie G un graf conex și x și y două vîrfuri distincte ale lui G . Să se arate că există un lanț de extremități x și y care utilizează o singură dată toate muchiile lui G , dacă și numai dacă x și y sînt singurele vîrfuri de grad impar ale grafului



Fig. 11.30

11.4. Arbori

Un graf conex și fără cicluri se numește *arbore*. În figura 11.30 este desenat un arbore cu 10 vîrfuri.

Se observă din aceasta figură că oricum am suprima o muchie a arborelui se obține un graf neconex care are două componente conexe. De asemenea, oricum am uni printr-o muchie două vîrfuri neadiacente ale unui arbore se creează un ciclu unic. De exemplu, dacă adăugăm muchia $[3, 4]$ apare ciclul $[2, 3, 4, 2]$, dacă adăugăm muchia $[5, 7]$ apare ciclul $[5, 1, 10, 7, 5]$, etc. Aceste proprietăți au loc pentru orice arbore, așa cum rezulta din teorema următoare.

Teoremă. *Următoarele afirmații sînt echivalente pentru un graf G :*

- 1) G este un arbore.
- 2) G este un graf conex minimal, adică G este conex și dacă îi suprimăm o muchie oarecare $[x, y]$ graful obținut devine neconex.
- 3) G este un graf fără cicluri maximal, adică G nu conține cicluri și dacă x și y sînt două vîrfuri neadiacente ale lui G , atunci graful obținut din G prin adăugarea muchiei $[x, y]$ conține un ciclu.

Demonstrație. Vom arăta că $1) \Rightarrow 2)$, $2) \Rightarrow 1)$, $1) \Rightarrow 3)$ și $3) \Rightarrow 1)$, ceea ce va demonstra echivalența celor trei proprietăți ale unui graf.

$1) \Rightarrow 2)$ Să presupunem că are loc 1), adică G este conex și fără cicluri. Trebuie arătat că are loc 2), adică prin suprimarea oricărei muchii $[x, y]$ se obține un graf neconex. Să presupunem prin reducere la absurd că graful G_1 obținut din G prin suprimarea muchiei $[x, y]$ este conex, deci există un lanț $L = [x, \dots, z, \dots, y]$ de extremități x și y . Dacă se repetă un vîrf z în șirul care îl definește pe L , adică

$$L = [x, \dots, z, t_1, t_2, \dots, t_k, z, \dots, y]$$

vom suprima din lanțul L una dintre aparițiile lui z împreună cu vîrfurile t_1, \dots, t_k obținînd un nou lanț L' de extremități x și y . Continuînd acest procedeu vom obține în final un lanț elementar de extremități x și y în graful G_1 . Acest lanț elementar împreună cu muchia suprimată $[x, y]$ formează un ciclu în arborele G , ceea ce contrazice definiția unui arbore. Deci are loc 2).

$2) \Rightarrow 1)$ Dacă G este un graf conex minimal, să presupunem prin reducere la absurd că G conține un ciclu $[x, z_1, \dots, z_k, y, x]$. Prin suprimarea muchiei $[x, y]$ a acestui ciclu se obține un nou graf G_1 . Graful G_1 este conex deoarece în orice lanț de la u la v în graful G care folosește muchia $[x, y]$ putem înlocui aceasta muchie prin lanțul $L = [x, z_1, \dots, z_k, y]$ care există în G_1 , obținînd un lanț de la u la v în graful G_1 . Deci G_1 este conex, ceea ce contrazice 2). Rezultă că G este fără cicluri și este conex, deci G este arbore.

$1) \Rightarrow 3)$ Dacă G este arbore rezulta că G este fără cicluri. Fie x și y două vîrfuri neadiacente ale lui G . Am văzut că există un lanț elementar $[x, z_1, z_2, \dots, z_k, y]$ de extremități x și y . Deci prin adăugarea muchiei $[x, y]$ se obține un nou graf G_1 care conține ciclul $[x, z_1, \dots, z_k, y, x]$, adică are loc 3).

$3) \Rightarrow 1)$ Fie G un graf fără cicluri maximal. Trebuie arătat că G este conex. Presupunînd că G nu este conex, rezulta existența a două vîrfuri x și y care aparțin unor componente conexe diferite ale lui G . Prin adăugarea muchiei $[x, y]$ se formează un nou graf G_1 care nu poate conține un ciclu $[x, z_1, z_2, \dots, z_k, y, x]$, deoarece în acest caz G conține lanțul $[x, z_1, \dots, z_k, y]$, ceea ce contrazice pre-

supunerea că x și y aparțin unor componente conexe diferite ale lui G . Însă acest lucru contrazice proprietatea 3), deci G este conex. Deoarece G este fără cicluri rezultă că G este arbore și are loc 1). Demonstrația este încheiată.

Un graf parțial H al unui graf G cu proprietatea că H este arbore se numește *arbore parțial* al lui G .

Corolar. *Un graf G conține un arbore parțial dacă și numai dacă G este conex.*

Demonstrație. Pentru a demonstra necesitatea, sa observăm că deoarece arborele parțial H este conex și G se obține din H prin unirea prin muchii a unor vârfuri neadiacente din H , rezultă că și G este conex. Să arătăm acum că orice graf conex G conține un arbore parțial H . Dacă G este un graf conex minimal, din teorema precedentă rezultă că G este arbore, deci vom lua $H = G$. În caz contrar, există o muchie $[x, y]$ a lui G cu proprietatea că graful parțial G_1 obținut din G prin suprimarea muchiei $[x, y]$ este conex. Dacă G_1 este un graf conex minimal vom lua $H = G_1$. În caz contrar vom repeta pentru G_1 procedeul de suprimare a unei muchii aplicat lui G ș.a.m.d., pînă la obținerea unui graf conex minimal, care va fi un arbore parțial al lui G .

Procedeul descris are un număr finit de pași deoarece graful G de la care pornim are cel mult C_n^2 muchii dacă el are n vârfuri, iar la fiecare etapă se suprimă câte o muchie a grafului parțial obținut în acel moment.

Propoziția II.4.1. *Orice arbore cu $n \geq 2$ vârfuri conține cel puțin 2 vârfuri terminale (de gradul 1).*

Demonstrație. Să presupunem, prin reducere la absurd, că există un arbore A cu $n \geq 2$ vârfuri, care are cel mult un vîrf de gradul 1.

Fie $L = [v, z_1, \dots, z_k, y]$ un lanț elementar de lungime maximă al lui A , deci care conține un lanț maxim de muchii. Cel puțin una dintre extremitățile lui L , fie aceasta y , are gradul $d(y) \geq 2$ deoarece A are cel mult un vîrf de gradul 1. Deoarece y este adiacent cu z_k , rezultă că y mai este adiacent cu un vîrf al lui A . Deoarece lanțul L are o lungime maximă, rezultă că y nu poate fi adiacent decît cu unul dintre vîrfurile v, z_1, \dots, z_{k-1} , ceea ce produce un ciclu în graful A . Deoarece A este arbore am ajuns la o contradicție și proprietatea este demonstrată.

Să observăm că arborile din figura 11.30 are 10 vîrfuri și $9 = 10 - 1$ muchii. Această proprietate are loc în general, așa cum ne arată propoziția următoare.

Propoziția II.4.2. *Orice arbore cu n vîrfuri are $n - 1$ muchii.*

Demonstrația se face prin inducție după n . Pentru $n = 1$ există un singur arbore cu un vîrf și fără nici o muchie, pentru $n = 2$ obținem de asemenea un arbore unic A_2 cu 2 vîrfuri și o singură muchie.

Să presupunem că proprietatea este adevărată pentru orice arbore cu cel mult n vîrfuri și fie A un arbore cu $n + 1$ vîrfuri și m muchii. Conform propoziției precedente, arborele A are cel puțin două vîrfuri de gradul 1. Fie x unul dintre acestea și A_1 subgraful obținut din A prin suprimarea vîrfului x și a muchiei incidente cu x . Deoarece A nu conține cicluri, rezultă că nici A_1 nu conține cicluri. Vom arăta că în plus graful A_1 este conex, deci A_1 este un arbore cu n vîrfuri. Fie u și v două vîrfuri diferite între ele și diferite de x ale arborelui A . Deoarece A este conex rezultă că există un lanț, deci și un lanț elementar $L = [u, \dots, v]$. Dacă muchia incidentă cu x este $[x, y]$, rezultă că lanțul elementar L nu folosește muchia $[x, y]$ pentru că $d(x) = 1$. Deci L este un lanț de extremități u și v și pentru subgraful A_1 . Rezultă că A_1 este conex. Aplicînd ipoteza de inducție pentru A_1 găsim că el are un număr de muchii egal cu $m_1 = n - 1$. Dar A conține în plus față de A_1 muchia $[x, y]$, deci A are $m = (n - 1) + 1 = n$ muchii și proprietatea este demonstrată.

Proprietatea unui arbore de a fi un graf conex minimal face ca arborii să intervină într-o serie de probleme de optimizare, cum este următoarea:

Problema arborelui parțial minim

Să considerăm un graf conex $G = (X, U)$ și o funcție $c: U \rightarrow \mathbb{R}_+$ care asociază fiecărei muchii a grafului G un număr real pozitiv numit *costul* acelei muchii.

Costul unui graf parțial $H = (X, V)$ al lui G este egal prin definiție cu suma costurilor asociate muchiilor lui H , ceea ce vom nota prin:

$$c(H) = \sum_{u \in V} c(u).$$

Se pune problema determinării unui graf parțial H al lui G care să fie conex și să aibă un cost minim.

Un astfel de graf parțial de cost minim trebuie să fie un arbore parțial, deoarece arborii sînt singurele grafuri conexe minimale.

Intr-adevar, dacă H este un graf parțial de cost minim al lui G și H conține o muchie $u = [x, y]$ a carei suprimare conduce la un alt graf parțial H_1 , conex, rezultă că:

$$c(H) = c(H_1) + c(u) > c(H_1).$$

Însă inegalitatea obținută $c(H_1) < c(H)$ contrazice minimalitatea grafului H .

Rezultă ca orice graf parțial de cost minim care este conex este un arbore parțial al lui G . Dacă vîrfurile grafului G reprezintă de exemplu nodurile unei rețele de telecomunicații, iar costul unei muchii reprezintă costul instalării unei linii telefonice între cele 2 noduri ale rețelei reprezentate de extremitățile muchiei, problema determinării rețelei conexe de cost minim este tocmai problema găsirii unui arbore parțial de cost minim în graful care reprezintă rețeaua.

Rețeaua de comunicații obținută trebuie să fie conexa pentru a facilita realizarea de convorbiri telefonice între oricare doua noduri ale rețelei, direct sau indirect, printr-o serie de noduri intermediare.

Pentru găsirea unui arbore parțial de cost minim, pe care îl vom numi în continuare arbore parțial minim al unui graf conex, prezentăm următorul algoritm:

Algoritmul APM (arbore parțial minim)

Fînd dat un graf conex $G = (X, U)$ cu o funcție cost $c: U \rightarrow \mathbb{R}_+$, se alege o muchie u cu costul $c(u)$ minim. Dintre muchiile nealese se va selecta mereu muchia de cost minim care nu formează cicluri cu muchiile deja alese.

Aplicarea acestui algoritm se termină cînd se obține o mulțime de muchii V , deci un graf parțial $H = (X, V)$ al lui G cu $V \subseteq U$, cu proprietatea că oricare dintre muchiile rămase ale lui G formează cicluri cu muchiile lui H . Deci H este un graf fără cicluri, maximal, cu aceeași mulțime de vîrfuri ca G .

Conform teoremei demonstrate rezultă că H este un arbore parțial al lui G .

Să aplicăm algoritmul APM pentru graful conex G cu 6 vîrfuri din figura II 31, pentru care costurile muchiilor sînt desenate lîngă muchiile respective.

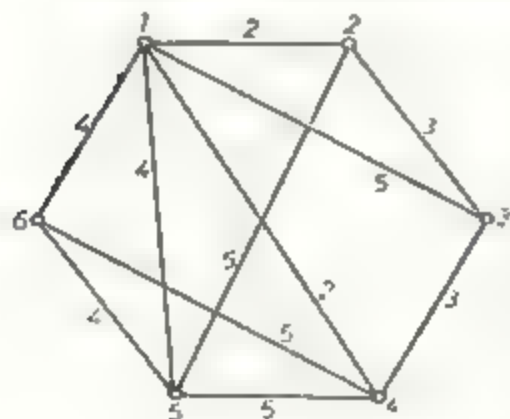


Fig. II 31

Alegem mai întîi o muchie de cost minim, de exemplu $[1, 2]$ cu costul 2, apoi muchia $[1, 4]$ care are de asemenea un cost minim egal cu 2. În continuare există două muchii dintre cele nealese avînd costul minim egal cu 3 unități și anume $[2, 3]$ și $[3, 4]$. O alegem de exemplu pe $[2, 3]$. Muchia rămasă de cost minim este $[3, 4]$, dar ea nu mai poate fi aleasă deoarece formează ciclul $[1, 2, 3, 4, 1]$ cu muchiile deja alese. Alegem de exemplu muchia $[1, 6]$ cu costul 4 și apoi muchia $[1, 5]$ cu costul 4, fără a apărea cicluri formate cu mu-

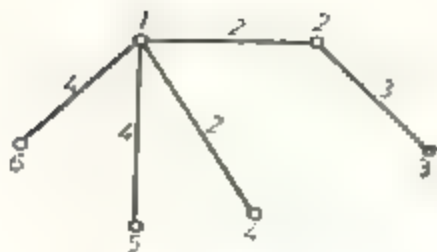


Fig. II.32

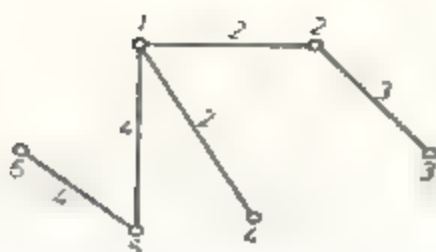


Fig. II.33



Fig. II.34

chile alese. Am obținut 5 muchii: $[1, 2]$, $[2, 3]$, $[1, 4]$, $[1, 6]$, $[1, 5]$, deci am obținut un arbore parțial al grafului G , deoarece propoziția II.4.2 ne arată că un arbore cu n vîrfuri are $n - 1$ muchii. Se observă ușor că oricum am adăuga o nouă muchie grafului parțial obținut în figura II.32 apare un ciclu.

Deci graful din figura II.32 este un arbore parțial minim, de cost egal cu 15, al grafului conex din figura II.31.

Dacă în locul muchiei $[1, 6]$ alegem muchia $[5, 6]$ cu același cost egal cu 4, obținem arborele minim din figura II.33. Se mai poate obține un arbore minim înlocuind muchia $[1, 5]$ prin muchia $[5, 6]$ pentru arborele din figura II.32. Se mai obțin trei arbori minimi dacă se înlocuiește, pentru fiecare din cei trei arbori obținuți, muchia $[2, 3]$ prin muchia $[3, 4]$ de același cost, egal cu 3. Unul dintre acești arbori este desenat în figura II.34. Rezultă că graful din figura II.31 are 6 arbori parțiali minimi.

În general, dacă există mai multe muchii cu același cost, pentru un graf conex G pot exista mai multe posibilități de alegere a unei muchii de cost minim care nu formează cicluri cu muchiile deja alese, deci pot exista mai mulți arbori parțiali minimi.

Vom încheia acest paragraf indicînd o formă a algoritmului *APM* care este ușor programabilă pentru un calculator electronic.

Pentru aceasta, să observăm că inițial se pleacă cu un graf parțial al grafului G care nu conține nici o muchie, deci care conține n vîrfuri izolate dacă G are n vîrfuri. Ulterior, prin adăugare de muchii se formează grafuri parțiale care nu conțin cicluri, deci care au drept componente conexe arbori. Se observă că o nouă muchie u poate fi selectată dacă are un cost minim printre muchiile nealese și dacă extremitățile ei aparțin unor componente conexe diferite ale grafului parțial obținut pînă în acel moment.

În caz contrar apare un ciclu, deoarece conform teoremei demonstrate un arbore este un graf fără cicluri, maximal.

Pentru a memora numerele de ordine ale componentelor conexe în care se găsesc la un moment dat vîrfurile grafului G vom folosi o listă cu n poziții, astfel încît poziția i din listă, notată cu $L(i)$, să indice numărul de ordine al componentei în care se găsește vîrfurile i al grafului.

Pentru a ușura căutarea muchiei de cost minim, vom alcătui lista muchiilor grafului G în ordine crescătoare a costurilor. Algoritmul se va opri după ce a selectat exact $n - 1$ muchii, deoarece un arbore cu n vîrfuri are $n - 1$ muchii.

Algoritmul *APM* devine:

1. Pentru $i = 1, \dots, n$ se face $L(i) \leftarrow i$;
2. Se alcătuiște lista muchiilor grafului G în ordinea crescătoare a costurilor;
3. Fie $[p, q]$ prima muchie din șirul muchiilor lui G ;
4. Au fost selectate $n - 1$ muchii? Dacă da, stop.
Am obținut un arbore parțial minim.
Dacă nu, mergi la pasul următor.

5. Se verifică dacă $L(p) = L(q)$. Dacă da, se consideră următoarea muchie din șirul de muchii ale lui G . Se notează cu p , respectiv q cele două extremități ale acestei muchii și se repeta pasul 5. Dacă $L(p) \neq L(q)$ se merge la pasul următor.

6. Se selectează muchia $[p, q]$ ca o nouă muchie a arborelui parțial minim. Dacă de exemplu $L(p) < L(q)$, toate elementele $L(i) = L(q)$ se înlocuiesc cu valoarea $L(p)$ și se merge la pasul 4.

Se observă că dacă $L(p) = L(q)$, cele două extremități ale muchiei $[p, q]$ sînt în același arbore, deci alegerea lui $[p, q]$ ar crea un ciclu în graful parțial obținut la momentul respectiv. Deci trebuie considerată următoarea muchie din șirul de muchii. La pasul 6 se unifică componentele conexe cărora le aparțin cele două extremități ale muchiei $[p, q]$, dînd tuturor vîrfurilor din reuniunea celor două componente numărul de ordine egal cu $L(p)$. La sfîrșitul aplicării algoritmului lista L va avea toate pozițiile egale cu 1. De fapt, după selectarea ultimei muchii putem să ne oprim, fără a mai unifica numerele de ordine ale celor două ultime componente conexe.

Pentru exemplul grafului din figura 11.31, o ordonare postula a muchiilor în ordinea crescătoare a costurilor este următoarea:

$[1, 2], [1, 4], [2, 3], [3, 4], [5, 6], [1, 5], [1, 6], [4, 5], [1, 3], [2, 5], [4, 6]$. Inițial, lista L

are forma $\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix}$. Deoarece $L(1) \neq L(2)$ se selectează muchia $[1, 2]$ și se

obține lista $L: \begin{bmatrix} 1 & 1 & 3 & 4 & 5 & 6 \end{bmatrix}$. Acum $L(1) = 1 \neq L(4) = 4$, deci alegem și muchia $[1, 4]$.

Noua listă $L: \begin{bmatrix} 1 & 1 & 1 & 1 & 5 & 6 \end{bmatrix}$. Obținem $L(2) = 1 \neq L(3) = 3$, deci alegem și muchia

$[2, 3]$, reactualizînd lista $L: \begin{bmatrix} 1 & 1 & 1 & 1 & 5 & 6 \end{bmatrix}$. Deoarece $L(3) = L(4) = 1$

nu vom alege muchia $[3, 4]$, trecînd la următoarea muchie $[5, 6]$. Deducem $L(5) = 5 \neq L(6)$

$= 6$, deci selectăm muchia $[5, 6]$ și găsim noua listă $L: \begin{bmatrix} 1 & 1 & 1 & 1 & 5 & 5 \end{bmatrix}$. Selectăm

și a V-a muchie $[1, 5]$ deoarece $L(1) = 1 \neq L(5) = 5$ obținînd arborele din figura 11.33 și noua listă L cu toate componentele egale cu 1.

Probleme

11.4.1. Să se găsească un arbore parțial minim al grafului conex din figura 11.35.

11.4.2. Fie G un graf cu $n \geq 3$ vîrfuri. Să se arate că următoarele afirmații sînt echivalente:

- G este un arbore;
- G este conex și are $n - 1$ muchii;
- G este fără cicluri și are $n - 1$ muchii.

11.4.3. Dacă G este un graf conex cu n vîrfuri, să se arate că pentru orice k astfel încît $1 \leq k \leq n$ există un subgraf conex al lui G cu k vîrfuri.

11.4.4. Să se arate că un graf cu n vîrfuri și cel puțin n muchii conține cel puțin un ciclu.

11.4.5. Fie G un graf conex cu n vîrfuri și m muchii. Să se arate că numărul minim de muchii care trebuie înlăturate astfel încît graful obținut să nu conțină cicluri este egal cu $m - n + 1$ (acest număr se numește numărul ciclomatic al grafului).

11.4.6. Numerele $d_1 \geq d_2 \geq \dots \geq d_n \geq 1$ sînt gradele vîrfurilor unui arbore cu $n \geq 2$ vîrfuri, dacă și numai dacă $d_1 + d_2 + \dots + d_n = 2n - 2$.

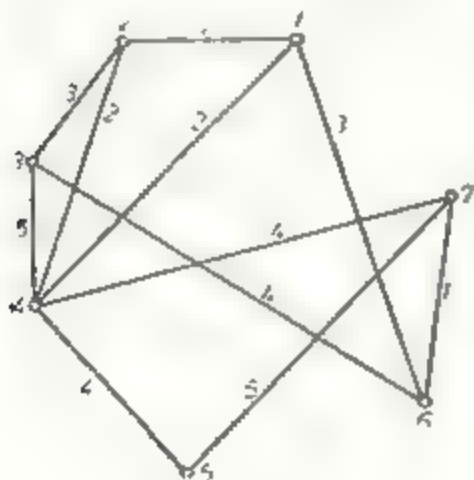


Fig. 11.35

11.5. Arbori binari și aplicații

Un *arbore binar* se definește în modul următor:

Este un arbore care are un vîrf numit *rădăcină*, al cărui grad este zero, unu sau doi. Dacă gradul rădăcinii este zero, arborele binar este format numai din rădăcină. În caz contrar, rădăcina se leagă printr-o muchie sau prin două muchii de unul sau două alte noi vîrfuri care se desenează sub rădăcină și care se numesc *fii* vîrfului rădăcină. Modul în care vîrfurile fii se desenează sub rădăcină, la stînga sau la dreapta, are importanță. Deci vom face distincție între fiul din dreapta și fiul din stînga rădăcinii. Aceste noduri fii au fiecare zero, unul sau două noduri fii, la stînga sau la dreapta ș.a.m.d. Vom spune că rădăcina arborelui are nivelul 0, fiii rădăcinii nivelul 1, fiii acestora nivelul 2, descendenții de ordin k ai rădăcinii nivelul k și îi vom desena la aceeași înălțime față de marginea de jos a unei pagini.

În figura 11.36 este reprezentat un arbore binar cu rădăcina A . Pe nivelul 1 apar vîrfurile B și C ; B este fiul din stînga al lui A și C este fiul din dreapta al lui A . Pe nivelul 2 se găsesc vîrfurile D și E , iar pe nivelul 3 vîrfurile F și G .

Vîrfurile terminale sau vîrfurile de grad 1 diferite de rădăcină ale acestui arbore sînt D, F, G, C . Se observa că aceste vîrfuri sînt vîrfurile care nu au fiu. În figura 11.37 sînt desenați doi arbori binari care sînt identici ca arbori, dar sînt distincți ca arbori binari, deoarece pentru unul dintre ei B este fiul din stînga al rădăcinii A , iar pentru celalalt B este fiul din dreapta al rădăcinii A .

Dacă suprimăm rădăcina și muchiile incidente cu aceasta, arborii obținuți se numesc *subarboarele stîng*, respectiv *subarboarele drept* al rădăcinii. Unul dintre aceștia sau amîndoi pot fi vizi. De exemplu pentru arborele binar din figura 11.36 vîrfurile A are subarboarele stîng format din vîrfurile B, D, E, F, G iar subarboarele drept format din vîrfurile C .

Noțiunea de subarbore se aplică și altui vîrf diferit de rădăcină.

De exemplu vîrfurile terminale D are ambii subarbori stîng și drept vizi, vîrfurile E are subarboarele stîng format din vîrfurile F și subarboarele drept format din vîrfurile G .

Dacă fiecare vîrf care nu este vîrf terminal al unui arbore binar are exact doi fii, arborele binar se numește *complet*.

Arborele binar din figura 11.36 este complet, dar arborii binari din figura 11.37 nu sînt compleți.

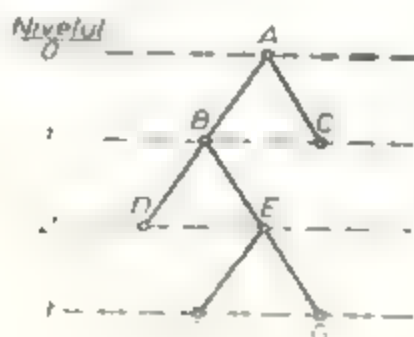


Fig. 11.36

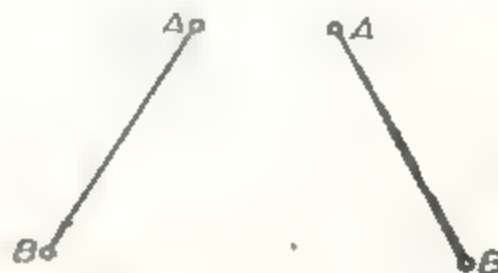


Fig. 11.37

11.5.1. Notăția fără paranteze a unei expresii aritmetice

Oricărei expresii aritmetice formate din constante și variabile și care utilizează operațiile de adunare, scădere, înmulțire, împărțire și ridicare la putere i se poate asocia un arbore binar. De exemplu, expresiei

$$(x + y)z + 7$$

i se asociază arborele binar complet din figura 11.38, unde semnul $*$ reprezintă înmulțirea.

În vîrfurile acestui arbore sînt reprezentați operatorii, adică operațiile din expresia aritmetică sau operanzii, adică constantele și variabilele din expresie. Ultima operație efectuată este adunarea, care se reprezintă în rădăcina arborelui binar. Subarborele stîng, respectiv subarborele drept reprezintă cele două expresii cărora li se aplica aceasta ultimă operație de adunare și anume $(x + y)z$ și 7 . Pentru expresia $(x + y)z$ se aplică aceeași regulă de reprezentare, deci în rădăcina subarborelui stîng vom reprezenta operația de înmulțire și a.m.d. Constanta 7 se reprezintă în rădăcina subarborelui drept, care nu mai conține alte vîrfuri.

Să reprezentăm după aceeași regulă expresia :

$$(x^2 - 3y)t + \frac{x}{y + 4}$$

Se obține arborele binar complet din figura 11.39, unde semnul $/$ reprezintă operația de împărțire, iar * reprezintă operația de ridicare la putere. Deci vom nota :

$$\frac{a}{b} = a/b, \text{ iar } a^b = a \uparrow b.$$

Un arbore asociat unei expresii aritmetice este un *arbore binar complet*, oricare vîrf neterminal avînd asociat un operator are exact doi fii. Într-adevăr, operațiile întîlnite într-o expresie aritmetică sînt operații binare, adică au doi operanzi. Vîrfurile terminale ale arborilor din figurile 11.38 și 11.39 au asociați operanzii din expresia aritmetică, care sînt constantele și variabilele expresiei.



Fig. 11.38

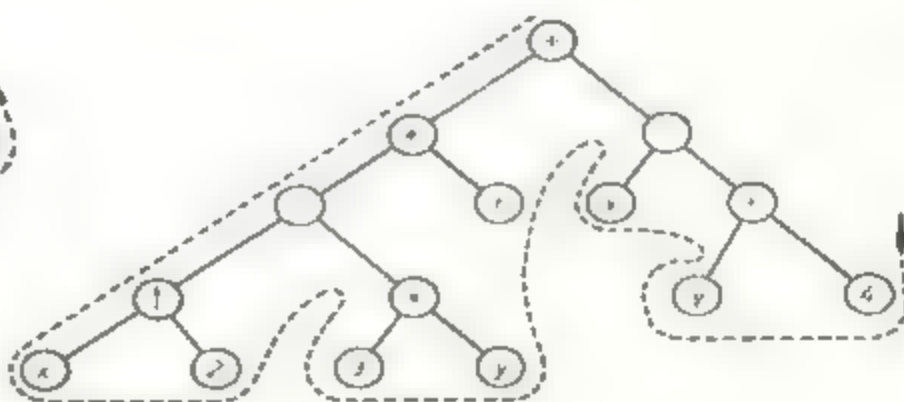


Fig. 11.39



Fig. 11.40

Se observă că deoarece operațiile de scădere, împărțire și ridicare la putere nu sînt comutative, arborii obținuți sînt într-adevăr niște arbori binari, pentru care se face distincție între fiul din dreapta și fiul din stînga al fiecărui vîrf neterminat.

Așa cum am făcut în figurile 11.38 și 11.39, vom reprezenta expresiile $a - b$, $\frac{a}{b}$ și a^b după cum se arată în figura 11.40, cu a desenat la stînga și b desenat la dreapta vîrfului care reprezintă operatorul de scădere, împărțire sau ridicare la putere.

În general, pentru a reprezenta o expresie aritmetică E sub forma unui arbore binar, procedăm după regula următoare :

a) Dacă E este o constantă sau o variabilă, arborele binar se reduce la rădăcina, careia îi asociem constanta sau variabila respectivă ;

b) Dacă $E = E_1 \alpha E_2$, unde α este unul dintre operatorii : $+$, $-$, $*$, $/$, \uparrow , vom asocia rădăcinii arborelui binar operatorul α , subarboarele stînga va reprezenta după aceeași regulă expresia aritmetică E_1 , iar subarboarele drept va reprezenta după aceeași regulă expresia E_2 .

Această regulă conduce, prin recurență, la un arbore binar complet asociat expresiei E . Invers, fiind dat un arbore binar complet care conține în nodurile terminale constante sau variabile și în celelalte noduri operatori, se poate obține ușor expresia aritmetică asociată. Din punctul de vedere al calculului automat, cu ajutorul unui calculator electronic, o altă formă a unei expresii aritmetice este mai utilă. Aceasta formă se obține de exemplu printr-un procedeu de *parcursere* a vîrfurilor arborelui, plecînd din rădăcina și trecînd prin fiecare vîrf o singură dată. Parcurserea arborelui binar asociat unei expresii aritmetice are loc după regula următoare : se pleacă din rădăcina pe muchia din stînga, iar în momentul cînd întreg subarboarele stînga al unui vîrf a fost parcurs, se parcurge subarboarele drept al aceluși vîrf după aceeași regulă. De fiecare dată cînd se trece printr-un vîrf se scrie operatorul sau operandul asociat aceluși vîrf.

Pentru arborii binari din figurile 11.38 și 11.39 ordinea de parcursere a fost desenată printr-o linie punctată.

De exemplu, pentru arborele din figura 11.38 plecăm din rădăcină pe muchia din stînga și ne deplasăm pe muchiile din stînga pînă ajungem în vîrf terminal asociat variabilei x . Deoarece subarboarele stînga al vîrfului diferit de rădăcină, asociat cu operația $+$, a fost parcurs, parcurgem subarboarele drept, care se reduce la rădăcină și are asociată variabila y . Acum subarboarele stînga al vîrfului asociat cu $*$ a fost parcurs, vom parcurge subarboarele drept, care se reduce la rădăcină și are asociată variabila z . Am parcurs subarboarele stînga al rădăcinii întregului arbore, deci mai rămîne de parcurs subarboarele drept care se reduce la rădăcină și are asociată constanta 7.

Am găsit succesiunea : $+ * + x y z 7$, care se numește *notația poloneză* sau *notația fără paranteze* a expresiei aritmetice $(x + y)z + 7$. Se folosește termenul de notație poloneză pentru șirul de semne obținut în urma par-

curgerii cu regula dată a unui arbore binar asociat unei expresii aritmetice, deoarece matematicianul polonez Lukasiewicz a fost primul care a definit-o.

În urma parcurgerii arborelui binar din figura II 39 se obține șirul de semne

$$+ * \quad \dagger \quad x \ 2 * 3 \ y \ \dagger / \ x \ \dagger \ y \ 4,$$

care reprezintă notația poloneză sau notația fără paranteze a expresiei aritmetice

$$(x^2 - 3y)\dagger + \frac{x}{y + 4}.$$

Pentru a calcula valoarea numerică a unei expresii aritmetice, trebuie ca toate variabilele să aibă atribuite anterior anumite valori numerice. Calculul valorii numerice a unei expresii aritmetice cu paranteze, pentru care toate variabilele au atribuite anumite valori numerice, se realizează într-un calculator, după ce a fost obținută notația poloneză a acelei expresii, în modul următor :

Se detectează în șirul de semne care alcatuiesc notația poloneză ultimul operator în sensul de la stînga la dreapta.

Se aplică acel operator celor două valori numerice care îl urmează în șir și numărul astfel obținut este plasat în șir în locul operatorului și a celor două numere care au fost utilizate în calcul.

Șirul nou obținut conține cu 2 semne mai puțin decît vechiul șir. Dacă am obținut un singur număr ne oprim, deoarece el reprezintă valoarea numerică a expresiei aritmetice. Dacă nu, aplicăm același procedeu șirului de semne obținut în acest moment.

În cazul șirului $+ * \dagger x \ y \ z \ 7$ se va înlocui succesiunea $+ x \ y$ prin rezultatul adunării valorii lui x cu valoarea lui y pe care îl notăm a . Se obține șirul : $+ * \dagger a \ z \ 7$, unde $a = x + y$. Se înmulțește a cu z și se obține șirul : $+ \dagger b \ 7$, unde $b = az$. În final se obține o singură valoare, $b + 7 = az + 7 = (x + y)z + 7$, adică valoarea numerică căutată.

Vedem deci cum un studiu de logică matematică și-a găsit aplicații la programarea calculatoarelor electronice, care au aparut cu 15 ani mai tîrziu.

II.5.2. Arbori de sortare

A sorta n numere reale a_1, a_2, \dots, a_n înseamnă a scrie aceste numere în ordine crescătoare, adică a găsi o permutare p a mulținii $\{1, \dots, n\}$ care verifică :

$$a_{p(1)} \leq a_{p(2)} \leq \dots \leq a_{p(n)}.$$

Dacă numerele a_1, \dots, a_n se afla depuse în aceeași ordine în memoria unui calculator, sortarea lor presupune rearanjarea acestor numere astfel încît ele să apară în ordine crescătoare.

Vom vedea cît de ușor putem găsi unul din aceste numere în cazul cînd ele sînt sortate, în raport cu situația cînd numerele sînt aranjate într-o ordine oarecare. Se apreciază ca cel puțin o treime din timpul de lucru al calculatoarelor care lucrează în domeniul informaticii de gestiune este afectat unor sortări de date.

Vom presupune mai întîr că vrem să determinăm cel mai mic dintre numerele a_1, \dots, a_n , pe care îl notăm $\min(a_1, \dots, a_n)$, comparînd numerele două cîte două.

Propoziție. Numărul minim de comparații a cite două numere necesare pentru a găsi $\min(a_1, \dots, a_n)$ este egal cu $n - 1$.

Demonstrație. Sa notăm numărul minim de comparații a cite două numere necesar pentru a găsi minimul din n numere prin $f(n)$.

Dacă $n = 2$ găsim minimul comparând cele două numere, deci $f(2) = 1$.

Presupunind proprietatea adevărată pentru orice $m \leq n - 1$, fie n numere a_1, \dots, a_n . După prima comparație găsim $b = \min(a_i, a_j)$, unde $1 \leq i < j \leq n$. Rămâne să determinăm minimul din $n - 1$ numere, care formează mulțimea $\{a_1, \dots, a_n\} \setminus \{a_i, a_j\}$, la care se adaugă numărul b .

Deci putem scrie:

$$f(n) = 1 + f(n - 1)$$

Deoarece conform ipotezei de inducție $f(n - 1) = n - 2$, rezultă că $f(n) = n - 1$ și proprietatea este demonstrată.

Procesul de găsim a minimului din n numere se poate reprezenta printr-un arbore binar. Într-adevăr, se poate proceda prin analogie cu reprezentarea unei expresii aritmetice printr-un arbore binar, deoarece operația de găsim a minimului din două numere este o operație binară.

De exemplu, dacă determinăm $\min(2, 7, 5, 1)$ putem proceda astfel: găsim $\min(2, 7) = 2$, apoi $\min(2, 5) = 2$ și în fine $\min(2, 1) = 1$.

Acest calcul este reprezentat prin arborele binar complet din figura 11.41.

În rădăcina arborelui apare numărul 1, care este rezultatul și fiecare vîrf neterminal are asociat un număr care reprezintă minimul celor două numere asociate fiilor săi.

Pentru a sorta n numere a_1, \dots, a_n putem proceda după cum urmează:

Determinăm mai întâi $\min(a_1, \dots, a_n)$ și acest număr îl punem pe primul loc în șirul sortat. Pentru a găsi numărul de pe locul al doilea determinăm minimul din cele $n - 1$ numere rămase ș.a.m.d.

Numărul de comparații necesare pentru a sorta cele n numere cu acest algoritm este egal cu:

$$(n - 1) + (n - 2) + \dots + 1 = \frac{n(n - 1)}{2}.$$

Vom vedea că acest număr poate fi îmbunătățit substanțial prin construcția unui arbore de sortare pentru care toate vîrfurile terminale apar pe un singur nivel sau pe două nivele consecutive. Pentru a construi un astfel de arbore, să observăm că dacă arborele este complet și toate vîrfurile terminale apar pe nivelul r , atunci el conține 2^r vîrfuri terminale.



Fig. 11.41

Într-adevăr, pe nivelul 1 sînt 2 vîrfuri, pe nivelul 2 sînt 4 vîrfuri și dacă presupunem că pe nivelul $r - 1$ sînt 2^{r-1} vîrfuri, rezultă că pe nivelul r sînt $2 \cdot 2^{r-1} = 2^r$ vîrfuri. Deci dacă numărul de vîrfuri terminale este o putere a lui 2, adică $n = 2^r$, putem construi un arbore binar complet cu toate vîrfurile terminale pe un același nivel.

În caz contrar, vom nota cu r numărul natural care verifică inegalitățile :

$$2^r < n < 2^{r+1}. \quad (1)$$

Să notăm cu x numărul de vîrfuri care se găsesc pe nivelul $r + 1$ într-un arbore binar complet cu toate vîrfurile terminale pe două nivele consecutive și cu y numărul de vîrfuri care se găsesc pe nivelul r .

Deoarece numărul total de vîrfuri terminale este egal cu n , putem scrie :

$$x + y = n. \quad (2)$$

Dacă construim pentru fiecare vîrf de pe nivelul r cei doi fii care se găsesc pe nivelul $r + 1$, obținem, împreună cu cele x noduri deja existente pe nivelul $r + 1$, un total de $2y$ vîrfuri pe nivelul $r + 1$. Deci mai obținem :

$$x + 2y = 2^{r+1}. \quad (3)$$

Ecuatiunile (2) și (3) formează un sistem care determină complet pe x și y și anume : $y = 2^{r+1} - n$ și $x = 2n - 2^{r+1}$. Deoarece sînt verificate inegalitățile (1) rezultă că $2^{r+1} - n > 0$ și $2n - 2^{r+1} > 0$, deci vîrfurile arborelui se găsesc pe nivelele r și $r + 1$. De exemplu, dacă $n = 6$ obținem $2^2 < 6 < 2^3$, deci $r = 2$ și $y = 2^3 - 6 = 2$ iar $x = 12 - 2^3 = 4$. Arborele binar complet cu 6 vîrfuri terminale pe două nivele consecutive va avea deci 2 vîrfuri pe nivelul 2 și 4 vîrfuri pe nivelul 3. El este desenat în figura 11.42. Să presupunem acum că vrem să sortăm numerele din șirul : 4, 6, 3, 5, 2, 8.

Vom aplica următorul algoritm, numit selecția arborească : reprezentăm aceste 6 numere în vîrfurile terminale ale arborelui din figura 11.42. Completăm celelalte vîrfuri interioare ale arborelui, astfel încît în fiecare vîrf interior să se afle scris minimul din cele două numere asociate fiilor săi.

Numărul de comparații necesare este egal cu 5, adică numărul minim de comparații pentru găsirea minimului din 6 numere.

În vîrfurile arborelui se află numărul 2, care este minimul din cele 6 numere. Scriem numărul 2 pe prima poziție a șirului sortat și scoatem numărul 2 din arborele de sortare, punînd o linioară în vîrfurile terminale unde a fost scris numărul 2. Pentru a completa arborele și a găsi minimul din numerele rămase, este suficient să completăm numerele înscrise în vîrfurile lanțului care unește rădăcina arborelui cu vîrfurile terminale unde a fost scris numărul 2.

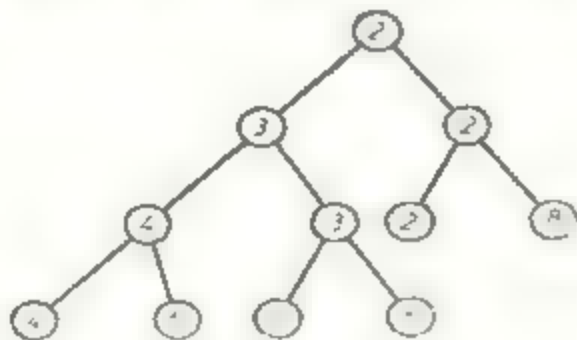


Fig. 11.42



Fig. 11.43

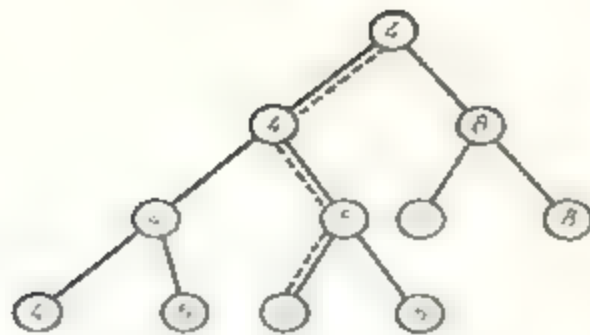


Fig. 11.44

Acest lanț a fost reprezentat cu linie punctată în figura 11.43. Numarul 8 nu are cu cine sa mai fie comparat pe nivelul 2, astfel încât el urcă pe nivelul 1. Aici el se compara cu 3. Gasim $\min(3, 8) = 3$, pe care îl scriem în rădăcina arborelui. Deci al doilea număr în șirul sortat este 3.

Considerind lanțul care trece prin vîrfurile cu numarul 3, reprezentat în figura 11.44, scoatem numărul 3 din arbore și mai facem două comparații: $\min(4, 5) = 4$ și $\min(4, 8) = 4$. În rădăcina arborelui apare acum numărul 4, care este al treilea număr din șirul sortat. Se reactualizează lanțul desenat punctat în figura 11.45 prin încă două comparații: $\min(5, 6) = 5$ și $\min(5, 8) = 5$, care se ridică în rădăcina arborelui. Al patrulea număr din șirul sortat este 5 și se obține arborele din figura 11.46 după încă o comparație: $\min(6, 8) = 6$. Deci penultimul număr din șirul sortat este 6. În final numărul 8 apare în rădăcina arborelui fără nici o comparație și el este ultimul număr din șirul sortat.

Să evaluăm numărul de comparații necesare pentru a sorta n numere, folosind algoritmul de selecție arborească.

Din (1) deducem că $r < \log_2 n < r + 1$, deci $r = \lfloor \log_2 n \rfloor$, adică r este partea întreagă din $\log_2 n$. Dacă $n = 2^r$ obținem că $r = \log_2 n = \lfloor \log_2 n \rfloor$, deoarece în acest caz $\log_2 n$ este număr întreg. La primul pas facem $n - 1$ comparații pentru a completa întregul arbore de sortare și deci pentru a găsi $\min(a_1, \dots, a_n)$ care apare în rădăcină. La fiecare din pașii următori facem comparații cel mult la nivelele $r, r - 1, \dots, 1$, pentru a completa vîrfurile care se găsesc pe lanțul care unește rădăcina cu vîrfurile terminale din care a fost suprimat un număr. Pentru exemplul dat, $\lfloor \log_2 6 \rfloor = 2$ și la fiecare pas ulterior am făcut cel mult $r = 2$ comparații.

Deci numărul de comparații este majorat de .

$$n - 1 + (n - 1)r = (n - 1)(r + 1) = (n - 1)(\lfloor \log_2 n \rfloor + 1).$$

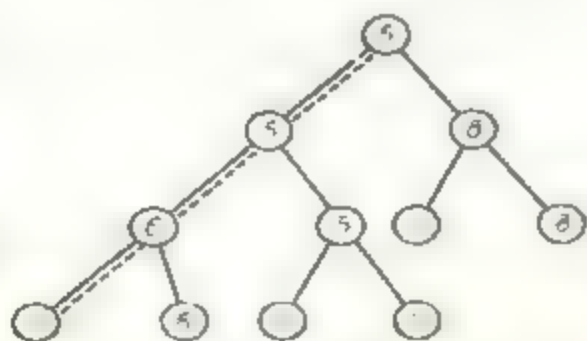


Fig. 11.45

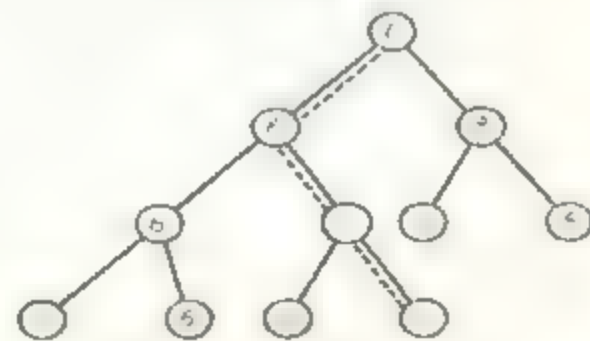


Fig. 11.46

Avantajul algoritmului selecției arborescente față de algoritmul de găsim repetată a minimumului în ceea ce privește numărul de comparații iese clar în evidență pentru n mare.

De exemplu, dacă avem de sortat 1 001 numere, cu algoritmul de găsim repetată a minimumului, trebuie să efectuăm

$$\frac{1\,001(1\,001 - 1)}{2} = 500\,500$$

comparații

Deoarece $2^9 = 512$, rezultă $r = \lceil \log_2 1\,001 \rceil = 10$ și algoritmul selecției arborescente necesită numai $1\,000 \cdot 10 = 10\,000$ comparații, adică de 50 de ori mai puțin.

Încheiem această aplicație menționând că există metode eficiente de reprezentare a structurii de arbore binar în memoria unui calculator.

11.5.3. Algoritmul de căutare binară

Problema regasirii informației stocate în memoria unui calculator constă în găsirea unui articol, care conține de obicei mai multe înregistrări memorate secvențial, după anumite valori, numerice sau alfanumerice, ale uneia sau mai multor înregistrări care formează acel articol.

Deci în cazul valorilor numerice, problema găsirii unui articol dintr-o mulțime de articole memorate secvențial în memoria unui calculator, revine în esență la următoarea problemă de căutare:

Dându-se n numere într-o ordine fixată, a_1, a_2, \dots, a_n , să se determine care număr are o valoare dată b . Vom presupune că problema are un rezultat unic, deci există cel mult un număr dintre a_1, \dots, a_n care este egal cu b . Putem rezolva această problemă citind pe rând fiecare dintre numerele a_1, a_2, \dots din memoria calculatorului și comparându-le cu b . În momentul cînd găsim un număr a_i egal cu b ne oprim sau citim de exemplu și celelalte înregistrări ale articolului care are înregistrarea căutată egală cu a_i . Spunem că în acest caz am avut o căutare cu succes. Dacă însă, după citirea tuturor numerelor a_1, \dots, a_n am constatat că nici unul dintre ele nu este egal cu b , spunem că a avut loc o căutare fără succes.

În cazul unei căutări cu succes numărul mediu de comparații necesar pentru găsirea valorii b este egal cu $\frac{n}{2}$, iar în cazul unei căutări fără succes numărul de comparații este egal cu n .

Dacă efectuăm multe căutări printre înregistrările a_1, \dots, a_n este mai avantajos din punctul de vedere al numărului de comparații, deci al vitezei de calcul, să se procedeze astfel: se sortează mai întâi cele n numere (de fapt articolele care conțin respectiv înregistrările a_1, \dots, a_n). Deci vom presupune în continuare că:

$$a_1 < a_2 < \dots < a_n$$

Apoi se aplică un algoritm rapid de căutare, care se bazează pe următoarea observație:

Dacă comparăm numărul a_i cu b , rezultă următoarele trei posibilități:

a) $a_i = b$ și căutarea este încheiată;

b) $a_i < b$ și deci vom continua căutarea printre numerele $a_{i+1} < a_{i+2} < \dots < a_n$, care sînt mai mari ca a_i . Dacă $i = n$, deci nu mai există numere mai mari ca a_i , ne oprim, deoarece căutarea nu a avut succes. Numarul b nu se găsește printre numerele a_1, \dots, a_n ;

c) $a_i > b$ și deci vom continua căutarea printre numerele $a_1 < a_2 < \dots < a_{i-1}$, care sînt mai mici ca a_i . Dacă $i = 1$, deci nu mai există numere mai mici ca a_i , ne oprim, deoarece căutarea nu a avut succes. Rezultă ca b nu se găsește printre numerele date. Algoritmul de căutare care va fi prezentat în continuare alege mereu numărul a_i la mijlocul șirului de numere în care căutam numărul b . De exemplu, dacă n este impar, numărul de indice $\frac{n+1}{2}$ se găsește la mijlocul șirului a_1, \dots, a_n . Dacă însă n este par, există două numere care se găsesc la mijlocul șirului și anume numerele de indici $\frac{n}{2}$ și $\frac{n}{2} + 1$. Deoarece n este par obținem $\frac{n}{2} = \left\lfloor \frac{n+1}{2} \right\rfloor$. Algoritmul de căutare binară utilizează mereu valoarea $\left\lfloor \frac{p+q}{2} \right\rfloor$ pentru indicele numărului de la mijlocul șirului:

$$a_p < a_{p+1} < \dots < a_q.$$

Algoritmul de căutare binară.

Fiind date numerele în ordine crescătoare $a_1 < a_2 < \dots < a_n$, se caută în acest șir numărul b .

1) Se stabilește $p \leftarrow 1, q \leftarrow n$.

2) $q < p$? Da: Stop. Algoritmul se termină fără succes. În caz contrar se stabilește $i \leftarrow \left\lfloor \frac{p+q}{2} \right\rfloor$.

3) Se compară b cu a_i . Dacă $b < a_i$, se merge la pasul 4); dacă $b > a_i$, se merge la 5), iar dacă $b = a_i$, algoritmul se termină cu succes.

4) Se atribuie $q \leftarrow i - 1$ și se merge la 2.

5) Se atribuie $p \leftarrow i + 1$ și se merge la 2.

La pașii 4 și 5 se redefinesc marginile subșirului care trebuie cercetat în continuare, în funcție de rezultatul comparației de la 3. Condiția de oprire în cazul căutării fără succes este $q < p$, așa cum vom verifica pe un exemplu.

Să aplicăm algoritmul de căutare binară pentru găsirea numărului 25 din șirul sortat:

$$3, 5, 12, 14, 15, 18, 24, 25, 42.$$

Deci $n = 9, a_1 = 3, a_2 = 5, \dots, a_8 = 25, a_9 = 42$. Inițial $p = 1, q = 9, i = 5$. La pasul 3 găsim $25 > a_5 = 15$, deci la pasul 5 redefinim $p = 6$. Se merge la 2 și se calculează $i =$

$$\left\lfloor \frac{15}{2} \right\rfloor = 7. \text{ La pasul 3 găsim } 25 > a_7 = 24, \text{ deci trecem la pasul 5 unde redefinim } p = 8.$$

Ne întoarcem la pasul 2 unde calculăm noua valoare a lui $i = \left\lfloor \frac{17}{2} \right\rfloor = 8$, iar la pasul 3 găsim $25 = a_8$. Algoritmul s-a terminat cu succes după trei comparații.

Să presupunem acum că vrem să căutăm (fără succes) numărul 13. Inițial $p = 1, q = 9, i = 5$. La pasul 3 găsim $13 < a_5 = 15$, deci la pasul 4 redefinim $q = 4$. Mergem la 2 și calculăm $i = \left\lfloor \frac{5}{2} \right\rfloor = 2$. La pasul 3 găsim $13 > 5 = a_2$, deci trecem la pasul 5 unde rede-

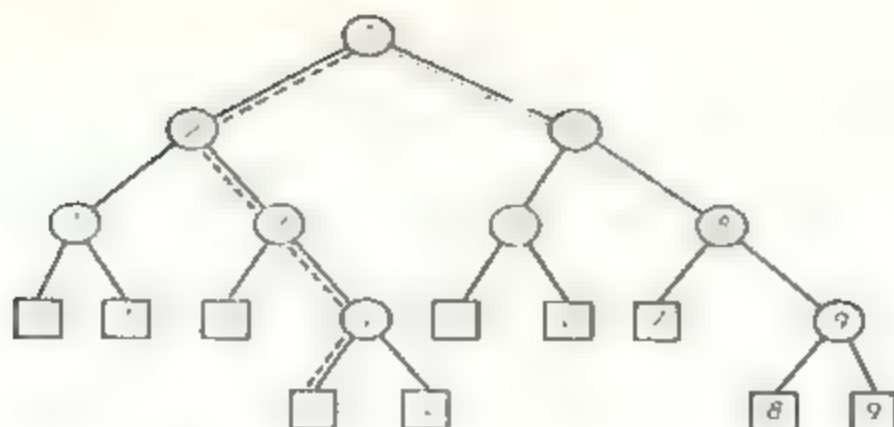


Fig. 11.47

avem $p = 3$. Ne întoarcem la 2 unde calculăm $i = \left\lfloor \frac{7}{2} \right\rfloor = 3$. La 3 găsim $13 > a_3 = 12$, deci mergem la pasul 5 unde obținem $p = 4$. Ne întoarcem la 2 cu valorile $p = q = 4$, deci $i = 4$, iar la 3 găsim $13 < a_4 = 14$. Mergem la 4 unde dăm valoarea $q = 3$.

Ne întoarcem la 2 unde găsim $q < p$, deci algoritmul s-a terminat fără succes după efectuarea a 4 comparații.

În cazul unei căutări fără succes a numărului b , la pasul 2 condiția $q < p$ este verificată când $p = r$, $q = r - 1$ și există inegalitățile:

$$a_{r-1} < b < a_r.$$

Într-adevăr, algoritmul localizează numărul b între numerele de indici $r - 1$ și r , deci la un moment dat $p = r - 1$ și $q = r$. Deoarece $b > a_{r-1}$, i se dă lui p valoarea r . Găsim $b < a_r$, deci q ia valoarea $r - 1$ și ne oprim deoarece $q < p$. Algoritmul de căutare binară poate fi privit ca un arbore de decizie binar, ca cel din figura 11.47 pentru cazul $n = 9$.

În acest caz prima comparație efectuată este aceea a lui b cu a_5 , care este reprezentată prin nodul rădăcină al arborelui binar din figura 11.47. Dacă găsim $b = a_5$ ne oprim. În caz contrar, dacă $b < a_5$ vom compara pe b cu a_3 , comparație care este reprezentată prin rădăcina subarborelui stâng al rădăcinii 5. Dacă însă $b > a_5$, vom compara pe b cu a_7 , comparație care este reprezentată prin rădăcina subarborelui drept al rădăcinii 5. În fiecare vîrf al arborelui care se obține în acest mod este scris indicele i al numărului a_i cu care se compară b în acel moment. Dacă $b < a_i$ se urmează ramura din stînga, efectuînd o comparație cu numărul al cărui indice este scris în rădăcina subarborelui stîng al nodului i . Dacă $b > a_i$ se urmează ramură din dreapta, efectuînd o comparație cu numărul al cărui indice este scris în rădăcina subarborelui drept al nodului i . O căutare fără succes va conduce la unul din nodurile terminale, reprezentate printr-un pătrat, numerotate de la 0 la 9 în sensul de la stînga spre dreapta.

De exemplu, se ajunge la nodul 0 dacă $b < a_1$, la nodul 1 dacă $a_1 < b < a_3$, ..., la nodul 9 dacă $b > a_9$.

Căutarea cu succes a numărului 25 a fost reprezentată în figura 11.47 prin linie punctată și corespunde parcurgerii lanțului care unește rădăcina arborelui cu nodul neterminal cu numărul 8. Căutarea fără succes a numărului $b = 13$ a fost reprezentată prin linie întreruptă și corespunde parcurgerii lanțului care unește rădăcina cu nodul terminal cu numărul 3, deoarece $a_3 < b < a_4$. Se observă că în fiecare caz numărul de comparații este egal

cu numărul de noduri neterminale care se găsesc pe lanțul care unește rădăcina cu vârful care reprezintă sfârșitul procesului de căutare. Astfel, în cazul căutării numărului 25 $= a_5$ s-au efectuat trei comparații, iar în cazul căutării fără succes a numărului $a_5 < 13 < a_4$ au fost efectuate patru comparații: cu a_5 , cu a_4 , cu a_3 și cu a_2 .

Reprezentarea strategiei de căutare în cazul algoritmului de căutare binară printr-un *arbore de decizie binar* ne va ajuta la evaluarea performanțelor acestui algoritm. Folosim termenul de arbore de decizie binară deoarece decizia de continuare a procesului de căutare, adică fie oprirea căutării, fie căutarea pe ramura din stînga, fie căutarea pe ramura din dreapta a unui nod, depinde de rezultatul comparației numărului căutat cu numărul din șir indicat de nodul în care ne aflăm. Evaluarea eficienței acestui algoritm este dată de următoarea teoremă, care poate fi demonstrată prin inducție după k .

Teoremă. Dacă efectuăm o căutare într-un șir ordonat de n numere cu algoritmul de căutare binară și dacă n verifică inegalitățile:

$$2^{k-1} \leq n < 2^k, \quad (1)$$

atunci:

- O căutare cu succes necesită cel mult k comparații.
- O căutare fără succes necesită $k - 1$ sau k comparații.

O ilustrare a acestor proprietăți pentru cazul $n = 4$ este dată în figura II 48.

Observăm că numărul maxim de comparații în cazul unei căutări cu succes este egal cu numărul de nivele pe care se găsesc nodurile neterminale, adică cu 3. Se vede din figura II 48 că acest număr maxim de comparații se atinge numai pentru găsirea ultimului număr a_4 .

În cazul unei căutări fără succes numărul de comparații este egal cu 2 sau cu 3, deoarece nodurile terminale se găsesc pe nivelele 2 și 3. Din figura II 48 se observă că în acest caz numărul de comparații egal cu 3 se atinge numai pentru vîrfurile terminale 3 și 4, deci în cazurile cînd $a_4 < b < a_5$ și respectiv $b > a_5$.

Din (1) deducem $k - 1 \leq \log_2 n < k$, deci $k - 1 = \lfloor \log_2 n \rfloor$ de unde $k = \lfloor \log_2 n \rfloor + 1$.

De exemplu, dacă vrem să căutăm un număr printre 1 000 de înregistrări citind înregistrare cu înregistrare și făcînd comparațiile respective, trebuie să facem în medie 500 de comparații în cazul unei căutări cu succes și 1 000 de comparații în cazul unei căutări fără succes.

Dacă însă înregistrările sînt sortate, în cazul folosirii algoritmului de căutare binară vom evalua numărul de comparații aplicînd teorema anterioară.

Deoarece $2^{10} = 1\,024$, obținem $k = \lfloor \log_2 1\,000 \rfloor + 1 = 10$. Deci orice căutare cu succes necesită între una și zece comparații, iar orice căutare fără succes necesită 9 sau 10 comparații.

S-a obținut o reducere importantă a numărului de comparații, deci implicit și a timpului de căutare, în raport cu cazul înregistrărilor nesortate.

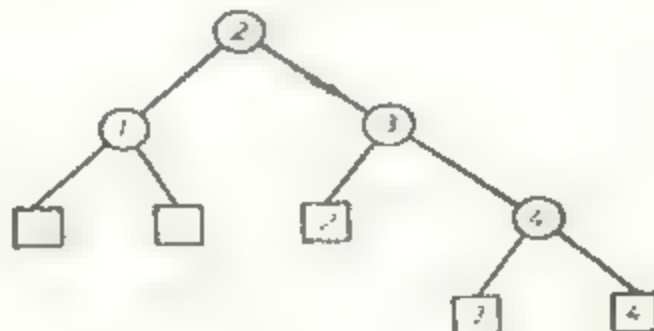


Fig. II 48

Probleme

11.5.1 Să se scrie notația fără paranteze pentru fiecare din următoarele expresii aritmetice :

a. $(x + y)(y + z)(z + x)$;

b. $3x^2y - 5x^2y^2 + 2xy^3$;

c. $\frac{x+y}{y+1} + \frac{y+z}{z+x} + \frac{z+x}{x+y}$,

d. $a^2 + b^2 + c^2 - 3abc$;

e. $(2x^{2y+z} - a)(14t - 3)$;

f. $a^{b^c} + \frac{x}{2 + \frac{y}{z}}$.

11.5.2. Să se arate că orice arbore binar complet cu n vârfuri terminale are în total $2n - 1$ vârfuri. Sa se deducă de aici o relație simplă între numărul de operatori și numărul de operanzi dintr-o expresie aritmetică.

11.5.3. Să se deducă scrierea obișnuită a următoarelor expresii aritmetice, date în notația fără paranteze, știind că toate constantele numerice utilizate au o singură cifră :

a. $- - * * * 5 x y z * 4 * \uparrow x 5 y * 3 z$

b. $/ x + y / \uparrow x 2 z$

c. $\uparrow x + a + * 2 b * 3 \uparrow c 2$

d. $* + * + x * 3 a - y 1 2 + \uparrow x 2 5$.

11.5.4 Să se construiască un arbore de sortare pentru cazul $n = 10$

11.5.5 Să se dea o definiție matematică pentru relația de ordine lexicografică (ordinea scrierii într-un dicționar) a cuvintelor formate cu litere dintr-un alfabet cu 24 de litere.

11.5.6 Pe o bandă magnetică sunt înscrise un milion de numere. Indicați un algoritm de determinare a numărului de numere distincte de pe bandă.

11.5.7 Se consideră un fișier cu 4 001 cuvinte binare distincte de 30 biți : $a_1, a_2, \dots, a_{4001}$. Două cuvinte binare $x = x_1 \dots x_{30}$ și $y = y_1 \dots y_{30}$ se numesc complementare dacă

$$x_i + y_i = 1 \text{ pentru } i = 1, \dots, 30.$$

Cu alte cuvinte, putem spune că x și y sînt complementare dacă și numai dacă

$$x + y = 11 \dots 1,$$

ele fiind considerate ca numere binare. Să se definească un algoritm pentru a găsi toate perechile de cuvinte complementare $\{a_i, a_j\}$, cu un număr cît mai redus de operații (de comparație și adunare).

11.5.8 La un turneu de tenis de cîmp participă n jucători. Cum trebuie organizat turneul, care conține numai meciuri eliminatorii de simplu, astfel încît dacă pentru determinarea campionului sînt necesare $n - 1$ meciuri, pentru determinarea celui de al II-lea jucător al turneului să mai fie necesară disputarea a numai $\lceil \log_2 n \rceil$ meciuri ?

11.5.9. Se consideră următorul algoritm de sortare prin interschimbare a șirului de numere a_1, a_2, \dots, a_n :

- 1 $i \leftarrow n$,
- 2 $t \leftarrow 0$. Efectuează pasul 3 pentru $j \leftarrow 1, 2, \dots, i-1$ și apoi execută pasul 4
- 3 Dacă $a_j > a_{j+1}$, interschimbă numerele a_j și a_{j+1} în șir (a_j este trecut pe poziția $j+1$, deci el va fi notat cu a_{j+1} și a_{j+1} trece pe poziția j , fiind notat în continuare cu a_j). Se stabilește $t \leftarrow j$.
- 4 Dacă $t = 0$ ne oprim. Șirul obținut este sortat. În caz contrar $i \leftarrow t$ și se revine la pasul 2.

Să se justifice acest algoritm, considerând mai multe exemple de aplicare și să se arate că are un număr finit de pași.

11.5.10 Se consideră două șiruri sortate de numere:

$$a_1 \leq a_2 \leq \dots \leq a_m \text{ și } b_1 \leq b_2 \leq \dots \leq b_n.$$

Să se propună un algoritm de *interclasare* a celor două șiruri de numere într-un singur șir sortat $c_1 \leq c_2 \leq \dots \leq c_{m+n}$, format din elementele celor două șiruri, care să folosească cel mult $m + n - 1$ comparații.

11.5.11 Ce algoritm de căutare corespunde arborelui binar complet din figura 11.49 în cazul unui șir sortat format din 4 numere?

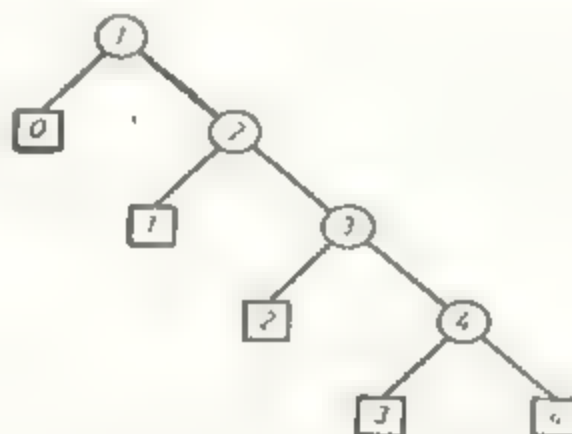


Fig. 11.49

11.5.12. Să se calculeze numărul mediu de comparații în cazul unei căutări cu succes și în cazul unei căutări fără succes pentru arborele de căutare din figurile 11.48 și 11.49.

11.5.13 O bandă magnetică conține n înregistrări.

Se presupune că frecvența acceselor la cele n înregistrări este diferită, adică unele sînt citite mai des, altele mai rar. Se știe că timpul de citire a unei înregistrări este cu atît mai mare cu cît înregistrarea respectivă se găsește mai departe de începutul benzii, deoarece crește timpul necesar derulării benzii magnetice.

Să se indice o strategie simplă de auto-organizare a datelor pe banda magnetică, astfel încît timpurile medii de acces la înregistrări să fie cît mai mici.

III. GRAFURI ORIENTATE

III.1. Noțiuni de bază

Un *graf orientat* G este format dintr-o pereche ordonată de mulțimi $G = (X, U)$. Ca și în cazul grafurilor neorientate, X este *mulțimea vîrfurilor* sau *nodurilor* grafului. Mulțimea U este formată din perechi ordonate de elemente distincte din X , numite *arce*. Orice arc $u \in U$ va fi notat prin $u = (x, y)$ cu $x, y \in X$ și $x \neq y$.

Spunem că vîrfurile x și y sunt *adiacente* în G și amîndoua sînt *incidente* cu arcul (x, y) . Deci un graf orientat G poate fi imaginat ca o mulțime de vîrfuri, dintre care unele sînt unite două cîte două prin arce. Un graf orientat poate fi desenat în plan reprezentînd vîrfurile sale prin puncte și arcele prin săgeți care sînt orientate de la extremitatea inițială către extremitatea finală a fiecărui arc.

Dacă graful G conține arcul (x, y) vom spune că vîrfurile x și y sînt *adiacente* în G și amîndoua sînt *incidente* cu arcul (x, y) . Deci un graf orientat G poate fi imaginat ca o mulțime de vîrfuri, dintre care unele sînt unite două cîte două prin arce. Un graf orientat poate fi desenat în plan reprezentînd vîrfurile sale prin puncte și arcele prin săgeți care sînt orientate de la extremitatea inițială către extremitatea finală a fiecărui arc.

Graful orientat $G = (X, U)$ unde:
 $X = \{1, 2, \dots, 8\}$ cu $U = \{(1, 2), (2, 3), (3, 1), (3, 2), (2, 4), (4, 5), (3, 5), (6, 8), (8, 7), (7, 8), (7, 6)\}$
 se reprezintă ca în figura III.1. Vom nota arcele așa cum se indică în figura, adică $u_1 = (1, 2)$, $u_2 = (3, 1)$, ..., $u_{11} = (6, 8)$.

Gradul exterior al unui vîrf x , notat prin $d^+(x)$, este numărul arcelor de forma (x, y) cu $y \in X$. *Gradul interior* al unui vîrf x , notat prin $d^-(x)$, este numărul arcelor de forma (y, x) cu $y \in X$.

De exemplu, pentru graful din figura III.1, obținem: $d^+(1) = 1$, $d^-(1) = 1$, $d^+(2) = 2$, $d^-(2) = 2$, $d^+(5) = 0$, $d^-(5) = 2$ etc.

Un *graf parțial* al unui graf orientat $G = (X, U)$ se definește în același mod ca și în cazul neorientat. El este un graf $G_1 = (X, V)$ unde $V \subset U$, deci este graful G însuși sau se obține din G prin suprimarea anumitor arce.

Și definiția unui *subgraf* al unui graf orientat $G = (X, U)$ este asemănătoare cu cazul neorientat. Prin definiție, un *subgraf* al lui G este un graf $H = (Y, V)$

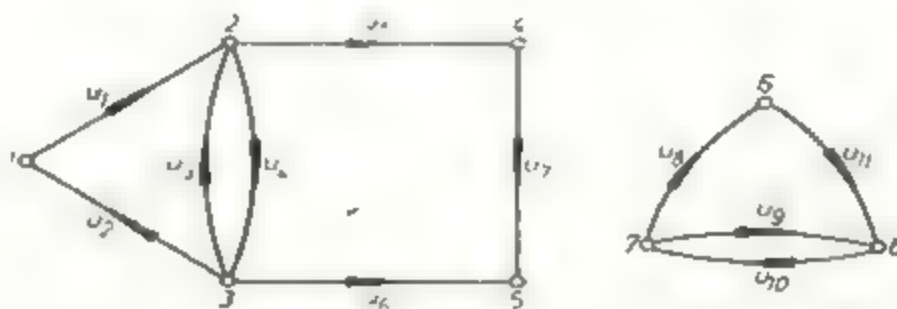


Fig. III.1

unde $Y \subset X$ iar arcele din V sînt toate arcele din U care au ambele extremități în mulțimea de vîrfuri Y .

Deci un subgraf H al unui graf orientat G este graful G însuși sau se obține din G prin suprimarea anumitor vîrfuri și a tuturor arcelor incidente cu acestea.

Vom spune că subgraful H este *indus* sau *generat* de mulțimea de vîrfuri Y .

Astfel, subgraful grafului G din figura III.1, indus de mulțimea de vîrfuri $Y_1 = \{1, 2, 4, 5\}$ are ca mulțime de arce mulțimea $V_1 = \{(1, 2), (2, 4), (4, 5)\}$, iar subgraful indus de mulțimea de vîrfuri $Y_2 = \{6, 7, 8\}$ are mulțimea arcelor $V_2 = \{(7, 6), (6, 8), (7, 8), (8, 7)\}$.

Un graf orientat este *complet* dacă oricare două vîrfuri sînt adiacente. De exemplu, subgraful grafului din figura III.1, indus de mulțimea de vîrfuri Y_2 , este complet.

În timp ce în cazul neorientat un graf complet cu n vîrfuri este unic determinat, în cazul orientat există mai multe grafuri complete cu un număr dat de vîrfuri. Ele se deosebesc fie prin orientarea arcelor, fie prin faptul că între două vîrfuri oarecare există un arc sau două arce de sensuri contrare.

*Un *lanț* al unui graf orientat se definește ca un șir de arce :

$$L = [u_1, u_2, \dots, u_p]$$

cu proprietatea că oricare două arce vecine u_i și u_{i+1} au o extremitate comună pentru orice $i = 1, \dots, p-1$.

Extremitatea x_0 a lui u_1 care nu este comună cu u_2 și extremitatea x_r a lui u_p care nu este comună cu u_{p-1} se numesc *extremitățile lanțului* L .

Dacă toate arcele lanțului L au o aceeași orientare, care este dată de sensul deplasării de la x_0 către x_r , lanțul se numește *drum*.

Deci un drum într-un graf orientat $G = (X, U)$ este un șir de vîrfuri notat :

$$D = (x_0, x_1, \dots, x_r)$$

cu proprietatea că $(x_0, x_1), (x_1, x_2), \dots, (x_{r-1}, x_r) \in U$, deci sînt arce ale grafului.

Vîrfurile x_0 și x_r se numesc *extremitățile drumului* D . Dacă vîrfurile x_0, x_1, \dots, x_r sînt distincte două cîte două, drumul D se numește *elementar*. Din aceste definiții rezultă că orice drum este și lanț, dacă îl privim ca un șir de arce.

Pentru graful din figura III.1 următoarele șiruri de arce sînt lanțuri :

$L_1 = [u_1, u_2, u_3, u_4]$, $L_2 = [u_4, u_5]$, $L_3 = [u_1, u_{10}]$, $L_4 = [u_9, u_1, u_2, u_3, u_4]$, $L_5 = [u_6, u_{11}]$. Lanțurile L_1 și L_5 sînt chiar drumuri și ele pot fi scrise ca un șir de vîrfuri în modul următor : $(2, 3, 2)$, respectiv $(7, 6, 8)$.

L_1 are extremitățile 1 și 4, L_5 are extremitățile 6 și 8, L_4 are extremitățile 2 și 1, iar pentru L_2 , cele două extremități coincid cu vîrful 2. Pentru același graf următoarele șiruri de vîrfuri sînt drumuri :

$D_1 = (1, 2, 4, 5)$, $D_2 = (7, 6, 8, 7)$, $D_3 = (3, 1, 2, 3, 5)$, $D_4 = (3, 2, 4, 5)$, $D_5 = (2, 3, 2, 3, 1, 2)$.

Drumurile D_1 și D_2 sînt drumuri elementare, deoarece nu trec de două ori printr-un același vîrf.

Un drum $D = (x_0, \dots, x_r)$ poate fi interpretat ca traseul unei deplasări pe arcele grafului în ordinea $(x_0, x_1), (x_1, x_2), \dots, (x_{r-1}, x_r)$.

De aceea drumul D de extremități x_0 și x_r se mai spune că este un drum de la x_0 la x_r . Dacă $x_0 = x_r$ și toate arcele $(x_0, x_1), (x_1, x_2), \dots, (x_{r-1}, x_r)$ sînt distincte două cîte două, drumul D se numește *circuit*.

Dacă toate vîrfurile circuitului, cu excepția primului și a ultimului vîrf, sînt distincte două cîte două, circuitul se numește *elementar*.



Fig. III 2

De exemplu drumul D_1 nu este circuit, deoarece folosește de două ori același arc a_1 . Drumul D_2 este un circuit elementar. Considerînd graful orientat din figura III 2, un circuit care nu este elementar este următorul $C = (1, 2, 1, 4, 3, 1)$, care trece de două ori prin vîrfurile 1.

Ca și în cazul ciclurilor din grafurile neorientate, circuitele pot fi scrise în mai multe moduri, alegînd ca prim vîrf un vîrf arbitrar prin care trece circuitul.

De exemplu, circuitul C din figura III 2 mai poate fi scris

$C = (2, 1, 4, 3, 1, 2)$ sau $C = (4, 3, 1, 2, 1, 4)$ sau

$C = (1, 4, 3, 1, 2, 1)$ sau $C = (3, 1, 2, 1, 4, 3)$

Noțiunile de *conexitate* și de *componentă conexă* a unui graf orientat sînt similare cu cele de la grafurile neorientate, utilizînd noțiunea de lanț din cazul grafurilor orientate.

Astfel, un graf orientat G se numește *conex* dacă pentru oricare două vîrfuri distincte x și y există un lanț de extremitați x și y în G . O *componentă conexă* C a unui graf orientat G se definește ca fiind un subgraf conex maximal al lui G , deci nu există nici un lanț care să unească un vîrf din C cu un vîrf care nu aparține lui C .

Graful din figura III 1 nu este conex și are 2 componente conexe C_1 indusă de mulțimea de vîrfuri $\{1, 2, 3, 4, 5\}$ și C_2 indusă de mulțimea de vîrfuri $\{6, 7, 8\}$.

Graful din figura III 2 este însă conex, deci are o singură componentă conexă.

Probleme

- III 1.1. Să se găsească drumurile elementare de la vîrfurile 1 la vîrfurile 8 pentru graful din figura III 3. Care sînt circuitele elementare ale acestui graf?

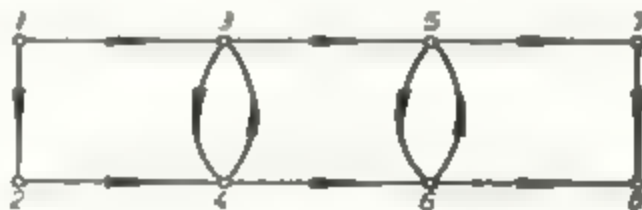


Fig. III 3

- III 1.2. Să se arate că dacă graful orientat G cu mulțimea de vîrfuri X are m arce, au loc egalitățile :

$$\sum_{x \in X} d^-(x) = \sum_{x \in X} d^+(x) = m.$$

- III 1.3. Să se calculeze numărul grafurilor orientate cu n vîrfuri date x_1, \dots, x_n . Cîte grafuri orientate și complete cu n vîrfuri date există?

- III.1.4 Pe mulțimea X a vîrfurilor unui graf orientat $G = (X, U)$ se introduce următoarea relație binară: Spunem că x este în relație cu y și scriem $x \sim y$ dacă $x = y$ sau dacă există un drum de la x la y și un drum de la y la x .
Să se arate că această relație binară este o relație de echivalență. Clasele acestei echivalențe se numesc *componentele tare conexe* ale grafului. Să se determine componentele tare conexe ale grafului din figura III.1.
- III.1.5 Fie $D = (x, \dots, y)$ un drum de la x la y ($x \neq y$) în graful G . Să se arate că există un drum elementar de la x la y în G .
- III.1.6 Un graf orientat cum este cel din figura III.4, care are proprietatea că între oricare două vîrfuri distincte x și y există un arc și numai unul se numește *graf turneu*.

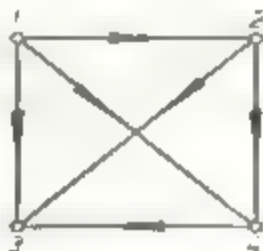


Fig. III.4

Un graf-turneu poate reprezenta rezultatele meciurilor directe între participanții la o competiție sportivă în care nu există meciuri nule și nici meciuri eliminatorii. Să se arate că orice graf-turneu conține un drum elementar care trece prin toate vîrfurile grafului.

- III.1.7. Pentru un graf turneu cu n vîrfuri x_1, \dots, x_n se notează $r_i = d(x_i)$ și $s_i = d^+(x_i)$ pentru $i = 1, \dots, n$. Să se arate că:

$$1) r_i + s_i = n - 1$$

$$2) \sum_{i=1}^n r_i = \sum_{i=1}^n s_i = C_n^2$$

$$3) \sum_{i=1}^n r_i^2 = \sum_{i=1}^n s_i^2$$

- III.1.8. Să se arate că pentru orice graf-turneu G există un vîrf x , astfel încît toate vîrfurile y diferite de x ale grafului pot fi atinse plecînd din x pe drumuri care au un arc sau două arce

III.2. Metoda drumului critic

III.2.1. Drumul critic într-un graf de activități

Una dintre aplicațiile cele mai răspîndite ale teoriei grafurilor în probleme de economie o constituie *metoda drumului critic*.

O lucrare complexă a cărei realizare comportă mai multe activități, se poate reprezenta printr-un graf orientat care dă o imagine grafică a eșalonării în timp și a intercondiționării tuturor activităților elementare care alcătuiesc întreaga lucrare.

Pentru un astfel de graf, arcele reprezintă *etapele sau activitățile elementare* ale lucrării. Fiecare arc are o anumită lungime, care reprezintă timpul de desfășurare a activității asociate acelui arc.

Momentul începerii activității este reprezentat de extremitatea inițială, iar momentul terminării activității este reprezentat de extremitatea finală a arcului respectiv.

Astfel pentru graful de activități din figura III 5 începerea întregii lucrări este reprezentată prin vârful A , iar terminarea ei prin vârful H .

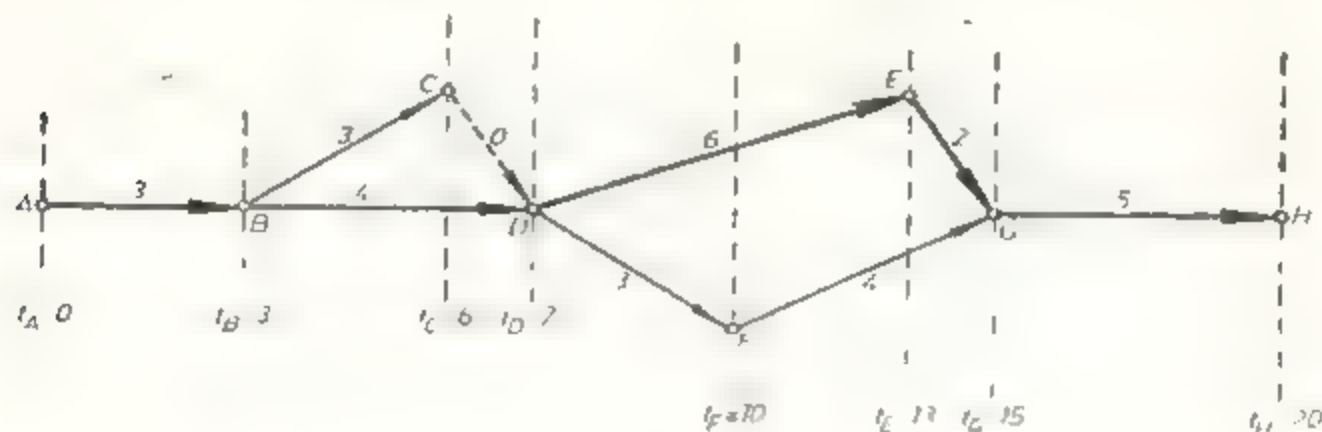


Fig. III 5

În această lucrare anumite activități se desfășoară în același timp, de exemplu activitățile reprezentate de arcele (B, C) și (B, D) .

La fel activitățile reprezentate de arcele (D, E) și (D, F) se desfășoară în același timp cu activitățile reprezentate de arcele (D, E) și (F, G) .

Nodurile în acest graf de activități reprezintă *evenimente*, care pot fi interpretate ca indicând realizarea unor obiective parțiale ale lucrării.

Astfel, vârful A reprezintă începerea întregii lucrări, vârful B reprezintă evenimentul care constă în terminarea activității reprezentate de arcul (A, B) și începerea activităților reprezentate de arcele (B, C) și (B, D) , nodul D reprezintă terminarea activităților reprezentate de arcele (C, D) și (B, D) și începerea activităților reprezentate de arcele (D, E) și (D, F) etc.

Activitatea reprezentată de arcul (C, D) , desenat cu linie întreruptă, este o activitate fictivă, care are asociat un timp egal cu zero și reprezintă pur și simplu o relație de anterioritate între terminarea operației (B, C) și începerea operațiilor reprezentate de arcele (D, E) și (D, F) .

La definirea grafului care reprezintă succesiunea în timp a activităților unei lucrări complexe, pentru orice vârf x trebuie să fie îndeplinită următoarea regulă:

Toate arcele care pleacă din x reprezintă operații care nu pot începe decât după terminarea tuturor operațiilor reprezentate de arcele care sosesc în vârful x .

Activitățile fictive se introduc pentru a nu avea mai multe arce paralele și de același sens între două vârfuri ale grafului.

Un graf de activități are o proprietate importantă și anume aceea că nu conține circuite, deoarece în caz contrar, conform regulii introduse, o aceeași operație ar trebui să înceapă după terminarea ei însăși, ceea ce este absurd. Într-adevăr, dacă într-un graf de activități, ar exista un circuit:

$$C = (x, y_1, y_2, \dots, y_p, x)$$

aceasta ar însemna că operația (y_p, x) începe după terminarea operației (y_{p-1}, y_p) . Însă operația (y_{p-1}, y_p) poate să înceapă numai după terminarea operației (y_{p-2}, y_{p-1}) ș.a.m.d. Găsim că operația (y_p, x) poate începe numai după terminarea operației (x, y_1) , ceea ce contrazice convenția de alcătuire a unui graf de activități.

Duratele operațiilor sau *timpii operatori*, care sînt numere reale nenegative, sînt asociați arcelor grafului, care reprezintă operațiile lucrării.

Astfel arcului (A, B) i se asociază timpul 3, arcului (B, C) timpul 3, arcului (B, D) timpul 4 etc., unitatea de măsură a timpului fiind ziua, săptămîna, luna etc.

Acești operatori timpi vor fi interpretați drept lungimi ale arcelor respective. Lungimea unui drum într-un graf de activități se definește ca fiind egală cu suma lungimilor arcelor care compun acel drum, deci este egală cu timpul necesar pentru executarea tuturor operațiilor cuprinse în drumul respectiv.

Stabilirea grafului de activități necesită pentru fiecare operație cunoașterea duratei sale cit și a operațiilor care o preced nemijlocit, deci toate relațiile de ordine temporală privitoare la aceasta, sau antecedentele obligatorii.

Pentru graful din figura 111.5 succesiunea în timp a operațiilor a fost marcată prin linii verticale întrerupte, care ne indică evoluția realizării în timp a întregii lucrări. Astfel evenimentul A are data $t_A = 0$, evenimentul B are data $t_B = 3$, evenimentul C are data $t_C = 6$ și evenimentul D are data $t_D = 7$, deoarece pentru producerea evenimentului D trebuie ca ambele activități (B, C) și (B, D) să fie terminate. Deci $t_D = \max(t_B + 4, t_B + 3) = \max(7, 6) = 7$. În continuare găsim $t_E = t_D + 3 = 10$, $t_F = t_D + 6 = 13$, $t_G = \max(t_E + 2, t_F + 4) = \max(15, 14) = 15$ și $t_H = t_G + 5 = 20$.

Rezultă ca timpul total de execuție a întregii lucrări este de 20 unități de timp.

Considerînd ca evenimentul care constă în începerea întregii lucrări are data zero, timpul total de execuție a lucrării este data realizării ansamblului de lucrări.

Această dată este egală cu suma timpilor operatori considerați pe drumul cel mai nefavorabil de la A la H , adică acel drum care dă între aceste două vîrfuri o sumă maximă de timpi operatori.

Acest drum, care poate să nu fie unic, este eluar drumul de lungime maximă de la A la H , numit și *drum critic*. Drumul critic pentru graful din figura 111.5 a fost desenat cu linii îngroșate și el este (A, B, D, E, G, H) . Drumul critic reunește activități de a căror realizare la timpul prevăzut depinde realizarea la timp a întregii lucrări.

Orice mărire a timpului asociat arcelor drumului critic conduce la mărirea timpului de realizare a întregii lucrări.

De aceea în cursul execuției lucrării activitățile situate pe drumul critic și numite activități critice, trebuie supravegheate cu deosebită grijă, pentru a nu depăși timpul planificat de realizare a lucrării.

Calculul datei de realizare a evenimentului final H revine deci la căutarea în graf a drumului cel mai lung sau a drumului critic de la A la H . Acest drum are o lungime de 20 unități de timp și am văzut că el reprezintă timpul de realizare a lucrării pornind de la data zero, care este atribuită evenimentului inițial A .

Deci dacă lucrarea se desfășoară fără incidente, durata sa va fi egală cu 20 unități de timp, de exemplu 20 de luni.

Operațiile critice sînt reprezentate de arcele (A, B) , (B, D) , (D, E) , (E, G) și (G, H) ale drumului critic.

III 2.2. Evenimente critice și intervale de fluctuație

Să considerăm un graf de activități G , ale carui vîrfuri reprezintă evenimentele E_1, E_2, \dots, E_n , vîrfuri care vor fi notate la fel cu evenimentele pe care le reprezintă. Presupunem că E_1 reprezintă începutul lucrării și E_n reprezintă

terminarea ei. Pentru a calcula datele de realizare ale diferitelor evenimente E_i , trebuie să determinăm cele mai lungi drumuri în graful de activități de la E_1 la celelalte vârfuri E_i , pentru a fi siguri că toate operațiile anterioare evenimentului E_i au fost terminate.

Lungimea maximă a drumurilor de forma $D = (E_1, \dots, E_i)$ din graful de activități se numește *data așteptată* a evenimentului E_i și se notează cu t_i .

Astfel pentru graful de activități din figura III 5 vom alege $t_A = 0$. Data așteptată a evenimentului B va fi $t_B = 3$, deoarece există un singur drum de la A la B și anume arcul (A, B) .

De la A la D există două drumuri, însă drumul (A, B, D) are lungimea maximă egală cu 7, deci $t_D = 7$. De la A la C există un singur drum de lungime 6, deci $t_C = 6$. La fel obținem $t_F = 10$ și $t_E = 13$. De la A la G există patru drumuri în graf și anume (A, B, C, D, E, G) ; (A, B, C, D, F, G) ; (A, B, D, F, G) și (A, B, D, E, G) , însă numai ultimul are o lungime maximă, egală cu 15. Deci $t_G = 15$. Am văzut că lungimea maximă a drumurilor de la A la H este egală cu 20, deci $t_H = 20$.

Este util să cunoaștem pentru fiecare eveniment E_i *data sa limită de realizare*, dată a carei depășire va determina întârzierea întregii lucrări. Timpul necesar pentru realizarea operațiilor situate între E_i și evenimentul final E_n , se obține cautând în graf drumul cel mai lung de la E_i la E_n , deoarece trebuie să fim siguri că operațiile care urmează după evenimentul E_i pot fi toate realizate, ținând seama de duratele lor.

Deci data limită a evenimentului E_i , care se notează t_i^* , se obține scăzând din data t_n a evenimentului final E_n lungimea maximă a drumurilor $D = (E_i, \dots, E_n)$.

Pentru graful de activități din figura III 5 obținem $t_H^* = t_H = 20$. Există un singur drum de la G la H și anume arcul (G, H) de lungime 5, deci $t_G^* = 20 - 5 = 15$. Deoarece drumurile (E, G, H) , respectiv (F, G, H) au o lungime egală cu 7, respectiv 9 obținem $t_E^* = 20 - 7 = 13$ și $t_F^* = 20 - 9 = 11$.

Cel mai lung drum de la D la H este (D, E, G, H) de lungime egală cu 13, deci $t_D^* = 20 - 13 = 7$.

În mod analog $t_C^* = 7$, $t_B^* = 3$ și $t_A^* = 0$. Pentru evenimentele critice A, B, D, F, G, H , se observă că aceste două date coincid.

$$t_A = t_A^* = 0, t_B = t_B^* = 3, t_C = t_C^* = 7, t_E = t_E^* = 13, t_G = t_G^* = 15 \text{ și } t_H = t_H^* = 20$$

Această proprietate are loc pentru toate evenimentele critice dintr-un graf de activități. Pentru a o justifica, să presupunem că evenimentul E_1 este critic, deci vârful E_1 se găsește pe un cel mai lung drum de la E_1 la E_n , fie $D = (E_1, \dots, E_i, \dots, E_n)$, a cărui lungime o notăm $l(D)$. În acest caz atât drumul $D_1 = (E_1, \dots, E_i)$ format cu șirul de vârfuri ale lui D dintre E_1 și E_i , cât și drumul $D_2 = (E_i, \dots, E_n)$ format cu șirul de vârfuri ale lui D dintre E_i și E_n , sînt drumuri de lungime maximă între extremitățile lor. Într-adevăr, dacă presupunem prin reducere la absurd că de exemplu D_1 nu are o lungime maximă, îl putem înlocui printr-un alt drum D_1' cu extremitățile E_1 și E_i , de lungime mai mare ca D_1 . Însă în acest caz drumul D_1' împreună cu D_2 formează un drum D' de la E_1 la E_n , de lungime $l(D') = l(D_1') + l(D_2) > l(D_1) + l(D_2) = l(D)$, deci $l(D') > l(D)$. Am ajuns astfel la o contradicție, deoarece am făcut ipoteza că D este un drum de lungime maximă de la E_1 la E_n . Am demonstrat astfel că atât D_1 cât și D_2 sînt drumuri de lungime maximă în mulțimea drumurilor de aceleași extremități. Rezultă ca $t_i = l(D_1)$ și $t_i^* = t_n - l(D_2)$. Deoarece $t_n = l(D) =$

$l(D_1) : l(D_2)$, deducem ca $t_i = t_i^* = l(D_1)$ pentru orice eveniment critic E_i , dintr-un graf de activități. Am obținut astfel pentru fiecare eveniment două date :

- data t_i este *data așteptată* a realizării evenimentului E_i ;
- data t_i^* este *data limită* de realizare a evenimentului E_i , după depășirea căreia timpul total de execuție a ansamblului lucrărilor este majorat.

Data t_i este numită și *data cea mai apropiată*, iar data limită t_i^* este numită și *data cea mai târzie* a evenimentului E_i .

Am văzut că pentru evenimentele critice data limită t_i^* se confundă cu data așteptată t_i . Intervalul de timp $[t_i, t_i^*]$ se numește *interval de fluctuație* și el reprezintă intervalul în care poate avea loc realizarea evenimentului necritic E_i , fara a modifica timpul total de execuție a ansamblului lucrărilor. În cazul evenimentelor critice acest interval se reduce la un singur număr, deoarece $t_i = t_i^*$.

Pentru exemplul considerat singurele evenimente necritice sînt C și F . Intervalul de fluctuație pentru C este $[6, 7]$, iar intervalul de fluctuație al evenimentului F este $[10, 11]$. Deci evenimentele C și F pot întârzia cu cel mult o unitate de timp de la data lor așteptată, fără ca data evenimentului final H să se modifice. Această întârziere poate avea loc de exemplu din cauza măririi timpilor operatori ai operațiilor (B, C) , respectiv (D, F) cu cîte o unitate.

Dacă ambele operații (B, C) și (D, F) își măresc durata de la 3 la 4 unități, în graful de activități toate operațiile devin critice, iar toate cele 4 drumuri de la A la H devin drumuri critice. În acest caz nu ar mai fi admisă nici o întârziere în executarea nici unei operații, pentru a nu produce întârzierea întregii lucrări.

Este util să cunoaștem pentru fiecare operație o_{ij} , reprezentată de arcul (E_i, E_j) în graful de activități, întârzierea care poate fi admisă la începerea sa, fara a modifica data așteptată de realizare a evenimentului E_j . Aceasta întârziere, numită *marginea liberă* a operației o_{ij} , este egală cu :

$$t_j - t_i - t_{ij}$$

unde t_i și t_j sînt datele așteptate ale evenimentelor E_i și E_j , care încadrează operația o_{ij} de durată t_{ij} .

Întârzierea care poate fi admisă la începerea operației o_{ij} , fara a modifica data limită de realizare a evenimentului E_j , se numește *marginea totală* a operației o_{ij} și ea este egală cu

$$t_j^* - t_i - t_{ij}$$

Aceasta este deci întârzierea maximă care poate fi admisă la începerea operației o_{ij} , fara a mări durata de execuție a întregii lucrări.

Într-adevar, operația o_{ij} are o durată egală cu t_{ij} , trebuie să se termine cel mai târziu la data t_j^* și nu poate începe mai devreme de data t_i , deci ea poate începe la cel mult $t_j^* - t_i - t_{ij}$ unități de timp după durată t_i . Deoarece $t_j^* \geq t_j$, rezultă că marginea totală este mai mare sau egală cu marginea liberă a oricărei operații.

Dacă operația o_{ij} se găsește pe drumul critic, deci este o operație critică, rezultă că evenimentele E_i și E_j sînt critice și în plus $t_j^* = t_j = t_i + t_{ij}$. Deci marginile libere, cît și marginile totale ale operațiilor critice sînt nule, începerea lor trebuind să se faca fara nici o întârziere.

Intervalele de fluctuație și marginile operațiilor măsoră elasticitatea unei lucrări, în sensul că, cu cît acestea sînt mai reduse, cu atît termenele de execuție a diferitelor faze ale lucrării sînt mai rigide.

În cazul în care timpii operatori ai operațiilor necritice pot fi măriți, marginea liberă va corespunde mării posibile a duratei unei operații necritice o_{ij} , lăsând neschimbate datele t_i și t_j ale evenimentelor E_i și E_j , care încadrează această operație.

Marginea totală a acestei operații va corespunde deci mării maxime a duratei operației o_{ij} , astfel încît evenimentul E_j să poată avea loc la data sa limită de realizare t_j^* , după depășirea căreia toată lucrarea ar întârzia, în condițiile în care evenimentul E_i se produce la data așteptată t_i .

De exemplu, pentru graful din figura III.5 marginea liberă a operației reprezentate de arcu (B, C) este

$$t_C - t_B - t_{BC} = 6 - 3 - 3 = 0.$$

În timp ce marginea sa totală este egală cu

$$t_C^* - t_B - t_{BC} = 7 - 3 - 3 = 1.$$

Deci operația (B, C) poate admite o întârziere maximă de o unitate de timp, fără a produce întârzierea execuției întregii lucrări.

Pentru operația necritică (D, F) marginea liberă este $t_F - t_D - t_{DF} = 0$ și marginea totală este egală cu $t_F^* - t_D - t_{DF} = 1$, iar pentru operația (F, G) marginea liberă este $t_G - t_F - t_{FG} = 1$ și marginea totală este $t_G^* - t_F - t_{FG} = 1$.

Cunoașterea intervalelor de fluctuație ale evenimentelor care compun o lucrare, precum și a marginilor totale ale operațiilor componente permite să se urmărească executarea la timp a întregii lucrări.

Astfel, dacă toate evenimentele se produc înaintea sau la datele lor limită t_i^* , lucrarea se va desfășura normal și data realizării finale t_n va fi respectată.

Dacă însă se va întâmpla ca data producerii unui eveniment E_i să depășească data sa limită t_i^* , pentru a nu se depăși data t_n de realizare a întregii lucrări va trebui să se accelereze execuția operațiilor situate pe cel mai lung drum de la E_i la E_n . Aceasta se poate face de exemplu prin alocarea de resurse suplimentare (forță de muncă, materiale sau utilaje) în detrimentul operațiilor programului care au margini totale mari.

Dacă nu există această posibilitate, se va putea evalua întârzieră datei t_n de realizare a ansamblului operațiilor care compun lucrarea.

Metoda drumului critic este utilizată, din aceste motive, în urmărirea execuției unor proiecte de cercetare sau de dezvoltare care pot cuprinde mii de operații, în urmărirea lucrărilor de construcții-montaj a unor obiective economice etc.

III.2.3. Determinarea drumului critic

Să presupunem că graful de activități G are vîrfurile x_1, \dots, x_n , unde x_1 reprezintă începerea lucrării, iar x_n reprezintă terminarea tuturor activităților lucrării. Vom nota prin t_{ij} durata activității reprezentate de arcu (x_i, x_j) al lui G .

Vom nota pentru orice vîrf x_i prin $p(x_i)$ predecesorii lui x_i , adică mulțimea vîrfurilor de la care pleacă arce care au extremitatea finală în x_i și prin $s(x_i)$ succesorii lui x_i , adică mulțimea vîrfurilor către care pleacă arce care au extremitatea inițială în x_i .

De exemplu, pentru graful de activități din figura III.6 obținem $p(1) = \emptyset$, $s(1) = \{2, 3\}$, $p(4) = \{2, 3\}$, $s(4) = \{5, 6\}$, $p(6) = \{4, 5\}$, $s(6) = \emptyset$ etc.



Fig. III.6

Vom arăta că datele t_i , respectiv t_i^* ale evenimentului reprezentat prin vârful x_i al grafului G se pot determina printr-un calcul recursiv, bazat pe formulele:

$$a) t_i = \max_{x_j \in p(x_i)} (t_j + t_{ij}),$$

unde toate datele t_j asociate vîrfurilor x_j care sînt extremitați inițiale ale arcelor (x_j, x_i) au fost calculate la un pas anterior, plecînd de la $t_1 = 0$, care este data începerii lucrării.

$$b) t_i^* = \min_{x_j \in s(x_i)} (t_j^* - t_{ij}),$$

unde toate datele t_j^* asociate vîrfurilor x_j care sînt extremitați finale ale arcelor (x_i, x_j) au fost calculate la un pas anterior, plecîndu-se de la $t_n^* = t_n$, care este data terminării lucrării.

Într-adevăr, am văzut că t_i este lungimea maximă a drumurilor de la x_1 la x_i în graful G . Dacă x_j este penultimul vîrf al unui drum de lungime maximă de la x_1 la x_i , putem scrie $t_i = t_j + t_{ij}$, deoarece drumul considerat de lungime maximă de la x_1 la x_i se compune dintr-un drum de lungime maximă de la x_1 la x_j , de lungime t_j , și din arcul (x_j, x_i) de lungime t_{ij} . Dacă însă $x_j \in p(x_i)$ și x_j nu este penultimul vîrf al nici unui drum de lungime maximă de la x_1 la x_i , rezultă $t_i > t_j + t_{ij}$, de unde se deduce formula a)

Data limită t_i^* se obține scăzînd din data t_n a evenimentului final lungimea maximă a drumurilor de la x_i la x_n .

Fie $D = (x_i, x_j, \dots, x_n)$ un drum de la x_i la x_n în graful G , unde $x_j \in s(x_i)$.

Să notăm prin D_j drumul care se obține din D prin suprimarea primului vîrf x_i . Este clar că D_j este un drum de la x_j la x_n în graful G și avem $l(D) = t_{ij} + l(D_j)$.

Putem deci scrie:

$$\begin{aligned} t_i^* = t_n - \max_j l(D) &= t_n - \max_j (t_{ij} + \max_{D_j} l(D_j)) = \min_j (t_n - \max_{D_j} l(D_j) - t_{ij}) = \\ &= \min_j (t_j^* - t_{ij}), \end{aligned}$$

unde j parcurge mulțimea indicilor cu proprietatea că $x_j \in s(x_i)$, adică există un arc de la x_i la x_j , deoarece orice drum $D = (x_i, x_j, \dots, x_n)$ de lungime maximă se compune din arcul (x_i, x_j) și un drum de lungime maximă de la x_j la x_n , iar $t_n - \max_{D_j} l(D_j) = t_j^*$.

Să aplicăm aceste formule pentru graful de activități din figura III.6. Mai întîi vom calcula datele t_i cu formula a). Plecăm cu $t_1 = 0$. Deoarece $p(2) = \{1\}$ și $p(3) = \{1\}$, rezultă $t_2 = t_1 + t_{12} = 2$ și $t_3 = t_1 + t_{13} = 3$. Avem $p(4) = \{2, 3\}$, deci vom calcula

$$t_4 = \max(t_2 + t_{24}, t_3 + t_{34}) = \max(7, 9) = 9.$$

În continuare $p(5) = \{3, 4\}$, deci

$$t_5 = \max(t_3 + t_{35}, t_4 + t_{45}) = \max(8, 13) = 13.$$

În fine data terminării lucrării este

$$t_6 = \max(t_4 + t_{46}, t_5 + t_{56}) = \max(13, 15) = 15, \text{ deoarece } p(6) = \{4, 5\}$$

Să calculăm acum datele limită t_i^* cu formulele b), plecând de la evenimentul final către evenimentul inițial.

Avem $t_6^* = t_6 = 15$. Următorul vîrf care va fi luat în considerație este vîrful 5, deoarece $s(5) = \{6\}$. Găsim

$$t_5^* = t_6^* - t_{56} = 13. \text{ Apoi } s(4) = \{5, 6\}, \text{ deci}$$

$$t_4^* = \min(t_5^* - t_{45}, t_6^* - t_{46}) = \min(11, 9) = 9.$$

În continuare putem alege fie vîrful 2, fie vîrful 3. De exemplu $t_2^* = t_4^* - t_{24} = 4$, deoarece $s(2) = \{4\}$. Apoi găsim:

$$t_3^* = \min(t_4^* - t_{34}, t_5^* - t_{35}) = \min(3, 8) = 3$$

Ca verificare obținem

$$t_1^* = \min(t_3^* - t_{13}, t_2^* - t_{12}) = \min(0, 2) = 0.$$

Evenimentele pentru care $t_i = t_i^*$ vor fi tocmai evenimentele critice din graful activităților. În cazul nostru găsim că toate evenimentele, cu excepția evenimentului 2, sînt critice.

Să observăm că dacă o activitate reprezentată de arcul (x_i, x_j) are ambele evenimente care o delimitează critice, adică $t_i = t_i^*$ și $t_j = t_j^*$, nu rezultă că această activitate este critică, decât dacă $t_j = t_i + t_{ij}$.

De exemplu, activitatea (3, 5) din graful din figura III.6 are ambele extremități evenimente critice, însă nu este o operație critică.

Într-adevăr, ea are o margine totală egală cu $t_5^* - t_3 = t_{35} = 5$ unități, deci execuția ei poate aduce o întârziere de cel mult 5 unități de timp, fără a afecta data de realizare a întregii lucrări. În schimb operația (3, 4) este critică deoarece $t_4 = t_3 + t_{34}$.

Deci activitățile critice ale grafului activităților se determină considerînd toate arcele cu ambele extremități evenimente critice și care în plus verifică condiția menționată.

Arcele critice compun drumul sau drumurile critice ale grafului activităților. În cazul exemplului considerat se obține un singur drum critic și anume (1, 3, 4, 5, 6). Cunoșcînd datele t_i și t_i^* se pot determina intervalele de fluctuație ale evenimentelor, cîi și marginile libere și totale ale activităților.

Se poate arăta că formulele a) și b) pot fi efectiv aplicate, deoarece graful activităților G nu conține circuite.

Dacă mulțimea A a vîrfurilor pentru care am determinat datele așteptate conține k elemente, pentru găsirea nouă date t_i a vîrfului $x_i \in A$ cu formula a) sînt necesare cel mult k adunări și $k-1$ comparații, acest număr maxim fiind atins cînd $p(x_i) = A$.

Deci, numărul de comparații necesare pentru a aplica formula a) este majorat de

$$\sum_{k=1}^{n-1} (k-1) = \sum_{k=1}^{n-2} k = \frac{(n-1)(n-2)}{2},$$

iar numărul de adunări este majorat de

$$\sum_{k=1}^{n-1} k = \frac{n(n-1)}{2}.$$

Obținem un rezultat analog pentru numărul de comparații și de scăderi necesitat de aplicarea formulei b). Deci pentru determinarea intervalelor de fluctuație ale celor n evenimente ale unui program prin acest algoritm sînt necesare în total cel mult $(n-1)(n-2)$ comparații și cel mult $n(n-1)$ adunări și scăderi.

Probleme

III.2.1 Să se determine intervalele de fluctuație și drumul critic pentru graful de activități din figura III.7

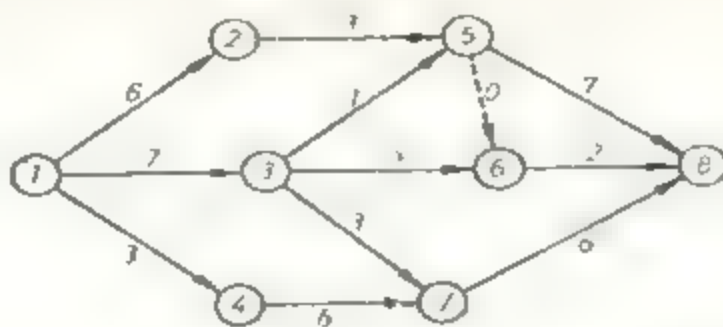


Fig. III.7

III.2.2. Să se construiască graful programului și să se determine drumul critic, intervalele de fluctuație ale evenimentelor și marginile totale ale operațiilor în următoarea situație de construcție a unui ansamblu hidroelectric, cu unitatea de timp luna. Operațiile care trebuie să fie realizate și duratele lor sînt următoarele.

- Construcția drumurilor de acces la uzină și la carierele pentru obținerea materialelor ($t_k = 4$);
- Pregătirea carierelor de exploatare și a fundațiilor ($t_k = 6$);
- Construcția unui centru pentru personal și administrație ($t_k = 4$);
- Comanda și construcția materialului electric și hidraulic (generatoare, turbine, conductă forțată etc.) ($t_k = 12$);
- Construcția uzinei ($t_k = 10$);
- Construcția barajului, a digurilor și a devierului de suprafață ($t_k = 24$);
- Construcția galeriilor de fuga și a conductelor de aducțiune ($t_k = 7$);
- Montajul uzinei și al conductelor ($t_k = 10$);
- Probele tehnologice ($t_k = 3$).

Evenimentele programului sînt următoarele:

- Pornirea lucrărilor;
- Terminarea drumurilor de acces;
- Terminarea centrului pentru personal și administrație și a fundațiilor;
- Terminarea uzinei și a montării conductelor de aducțiune;
- Terminarea ansamblului energetic;
- Livrarea lucrărilor către beneficiar.

III.2.3 Să se arate că orice graf orientat care nu conține circuite are cel puțin un vîrf x cu proprietatea $p(x) = \emptyset$ și cel puțin un vîrf y astfel încît $s(y) = \emptyset$.

III.2.4 Se consideră graful de activități din figura III.8, pentru care duratele operațiilor au fost scrise lângă arcele respective.

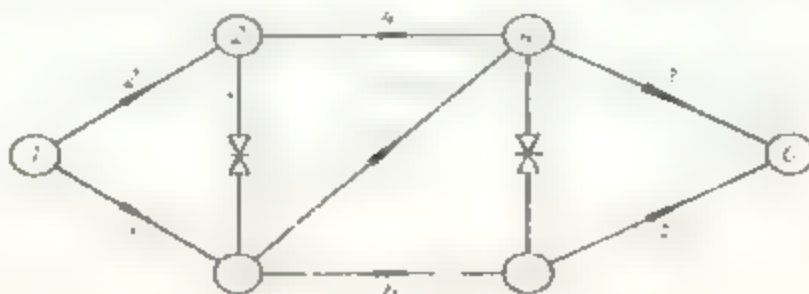


Fig. III.8

Operațiile reprezentate de arcul dintre vîrfurile 2 și 3 și arcul dintre vîrfurile 4 și 5 se numesc *operații disjunctive*, deoarece desenul din figură are următoarea interpretare: Există fie arcul (2, 3) de durată egală cu 4, fie arcul (3, 2) de durată egală cu 6. La fel pentru perechea de vîrfuri 4 și 5 există fie arcul (4, 5) de durată 2, fie arcul (5, 4) de durată 3, însă nu amîndouă aceste arce. Să se determine orientările arcelor disjunctive astfel încît întreaga lucrare să se termine într-un timp cît mai scurt, adică drumul critic să aibă o lungime minimă

III.3. Flux maxim într-o rețea de transport

Un graf orientat $G = (X, U)$ se numește *rețea de transport* dacă satisface următoarele condiții:

- a) există un vîrf unic $a \in X$ în care nu intră nici un arc sau $\omega^-(a) = \emptyset$, unde prin $\omega^-(a)$ se notează mulțimea arcelor care intra în vîrfurile a ;
- b) există un vîrf unic $b \in X$ din care nu iese nici un arc sau $\omega^+(b) = \emptyset$, unde prin $\omega^+(b)$ se notează mulțimea arcelor care ies din vîrfurile b ;
- c) G este conex și există drumuri de la a la b în G ;
- d) s-a definit o funcție $c: U \rightarrow \mathbb{R}$ astfel încît $c(u) \geq 0$ pentru orice arc $u \in U$.

Vîrfurile a se numește *intrarea rețelei*, vîrfurile b se numește *ieșirea rețelei*, iar numărul nenegativ $c(u)$ se numește *capacitatea arcului u .*

Un exemplu de rețea de transport cu 8 vîrfuri este reprezentat în figura III 9

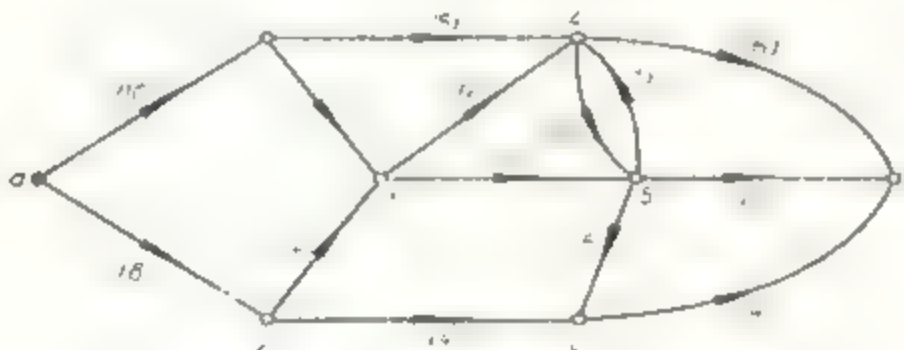


Fig. III 9

Vîrfurile a este intrarea rețelei, vîrfurile b este ieșirea rețelei și se verifică de exemplu că $\omega^-(a) = \emptyset$, $\omega^+(b) = \emptyset$, $\omega^-(b) = \{(4, b), (5, b), (6, b)\}$ și $\omega^+(a) = \{(a, 1), (a, 2)\}$.

Valorile capacității fiecărui arc au fost trecute în paranteze lângă arcele respective.

Graful din figura III 9 este conex și există drumuri de la a la b , de exemplu $(a, 1, 3, 5, b)$ sau $(a, 2, 3, 4, 5, b)$.

Un exemplu de lanț de extremități a și b care nu este drum este $[(a, 2), (2, 6), (5, 6), (5, b)]$.

O funcție $f: U \rightarrow \mathbb{R}$ astfel încît $f(u) \geq 0$ pentru orice arc u se numește *flux* în rețeaua de transport G cu funcția de capacitate c , care se notează $G = (X, U, c)$, dacă sînt îndeplinite următoarele două condiții:

C) Condiția de conservare a fluxului :

Pentru orice vîrf x cu $x \neq a$, $x \neq b$, suma fluxurilor pe arcele care intră în x este egală cu suma fluxurilor pe arcele care ies din x , adică :

$$\sum_{u \in \omega^-(x)} f(u) = \sum_{u \in \omega^+(x)} f(u)$$

pentru orice $x \in X \setminus \{a, b\}$.

Să observăm că această lege are aceeași formă cu prima lege a lui Kirchhoff din teoria rețelelor electrice.

M) Condiția de mărginire a fluxului :

Pentru orice arc al rețelei, valoarea fluxului nu poate depăși capacitatea arcului respectiv, adică

$$f(u) \leq c(u) \text{ pentru orice arc } u \in U.$$

Pentru rețeaua de transport din figura III 9 valorile unui flux pe arce au fost reprezentate în figura III 10, lângă numărul din paranteze care reprezintă capacitatea arcului respectiv.

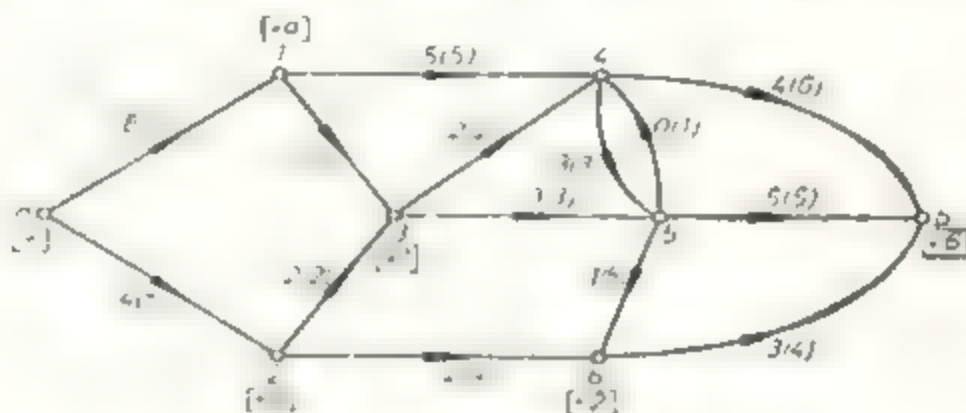


Fig. III.10

Se verifică condițiile C și M relative la vîrfurile și la arcele rețelei. De exemplu, pentru vîrful 4 condiția de conservare se scrie: $5 + 2 + 0 = 3 + 4$.

Pentru orice mulțime de vîrfuri $A \subset X$ vom defini o *tăietură de suport* A ca fiind mulțimea arcelor care intră în mulțimea A din exteriorul lui A și o vom nota astfel :

$$\omega^-(A) = \{(x, y) \mid x \notin A, y \in A \text{ și } (x, y) \in U\}.$$

Vom mai nota :

$$\omega^+(A) = \{(x, y) \mid x \in A, y \notin A \text{ și } (x, y) \in U\},$$

adică mulțimea arcelor care ies din mulțimea A de vîrfuri.

Capacitatea tăieturii $\omega^-(A)$ se notează prin $c(\omega^-(A))$ și ea este egală prin definiție cu $\sum_{u \in \omega^-(A)} c(u)$, adică cu suma capacităților arcelor care fac parte din tăietura considerată.

Pentru rețeaua de transport din figura III.9 dacă alegem $A = \{4, 3, 5, b\}$, obținem:

$$\omega^-(A) = \{(1, 4), (1, 3), (2, 3), (6, b)\} \text{ și}$$

$$\omega^+(A) = \{(5, 6)\}, \text{ iar capacitatea tăieturii de suport } A \text{ este } c(\omega^-(A)) = 5 + 7 + 2 + 4 = 18.$$

În continuare vom lucra numai cu tăieturi de această formă, avînd ca suport o mulțime de vîrfuri ale rețelei care conține ieșirea b și nu conține intrarea a .

Teorema III.3.1. *Fiind dată o rețea de transport cu intrarea a și ieșirea b și un flux f , are loc egalitatea:*

$$\sum_{u \in \omega^+(a)} f(u) = \sum_{u \in \omega^-(b)} f(u). \quad (1)$$

Demonstrație Egalitatea (1) exprimă faptul că fluxul care iese din a ajunge în b , deoarece este verificată condiția de conservare în fiecare vîrf $x \neq a, b$. Să calculăm în două moduri suma:

$$\sum_{x \in X} \left(\sum_{u \in \omega^-(x)} f(u) - \sum_{u \in \omega^+(x)} f(u) \right), \quad (2)$$

adică pentru fiecare vîrf al rețelei facem diferența dintre fluxul pe arcele care intră în acel vîrf și fluxul pe arcele care ies din acel vîrf și apoi însumăm toate aceste diferențe.

Din cauza condiției C de conservare a fluxului pentru orice vîrf diferit de intrare și de ieșire, termenul corespunzător din suma (2) este nul. Deci suma (2) se reduce la

$$\sum_{u \in \omega^-(b)} f(u) - \sum_{u \in \omega^+(a)} f(u), \quad (3)$$

deoarece $\omega^-(a) = \omega^+(b) = \emptyset$.

Însă fiecare arc $(x, y) \in U$ aparține atât mulțimii $\omega^+(x)$, cît și mulțimii $\omega^-(y)$, cînd valoarea fluxului $f(u)$ apare cu semne contrare, deci regroupînd termenii putem scrie suma (2) sub forma

$$\sum_{u \in U} (f(u) - f(u)) = 0. \quad (4)$$

Comparînd (3) cu (4) rezultă (1).

În continuare vom numi valoarea comună a celor două sume care apar în egalitatea (1) *fluxul la ieșirea rețelei* (care deci este egal cu fluxul la intrarea rețelei) și îl vom nota cu f_0 .

Pentru cazul fluxului reprezentat în figura III.10 fluxul la intrare este egal cu $8 + 4 = 12$, fluxul la ieșire este egal cu $f_0 = 4 + 5 + 3 = 12$, iar capacitatea tăieturii de suport $A = \{4, 3, 5, b\}$ am văzut că este egală cu $18 > 12$.

Proprietatea fluxului la ieșire, de a fi mai mic sau egal cu capacitatea oricărei tăieturi are loc în orice rețea de transport, așa cum ne indică teorema următoare

Teorema III.3.2. *Pentru o rețea de transport $G = (X, U, c)$ cu intrarea a și ieșirea b și un flux f , să considerăm o mulțime oarecare de vîrfuri $A \subset X$ cu proprietatea că $a \notin A$ și $b \in A$. Au loc relațiile:*

$$f_0 = \sum_{u \in \omega^-(A)} f(u) - \sum_{u \in \omega^+(A)} f(u) \leq c(\omega^-(A)) \quad (5)$$

Demonstrație Pentru a arăta că fluxul la ieșirea rețelei este egal cu diferența dintre suma fluxurilor pe arcele care intră în mulțimea A și suma fluxurilor pe arcele care ies din mulțimea de vîrfuri A , se poate proceda în mod analog ca pentru demonstrarea egalității (1), considerînd suma:

$$\sum_{x \in A} \left(\sum_{u \in \omega^{-}(x)} f(u) - \sum_{u \in \omega^{+}(x)} f(u) \right). \quad (6)$$

Deoarece pentru orice arc $u \in U$ există inegalitățile $0 \leq f(u) \leq c(u)$ putem scrie

$$f_b = \sum_{u \in \omega^{-}(A)} f(u) - \sum_{u \in \omega^{+}(A)} f(u) \leq \sum_{u \in \omega^{-}(A)} f(u) \leq \sum_{u \in \omega^{-}(A)} c(u) = c(\omega^{-}(A)).$$

În cele ce urmează vom studia problema determinării unui flux maxim f_b la ieșirea unei rețele de transport, pe care o vom numi pe scurt problema fluxului maxim.

Algoritmul lui Ford Fulkerson pentru obținerea unui flux maxim.

Pentru obținerea unui flux maxim la ieșirea b a unei rețele de transport $G = (X, U, c)$, unde capacitatea $c(u) \geq 0$ a fiecărui arc este un număr întreg, se procedează după cum urmează:

1) Se pleacă de la un flux inițial care verifică condițiile de conservare în fiecare vîrf și de mărginire pe fiecare arc, de exemplu de la fluxul avînd componente nule pe fiecare arc al rețelei.

Deci putem lua $f(u) = 0$ pentru orice arc $u \in U$.

2) Se determină lanțurile nesaturate de la a la b (adică lanțurile pe care fluxul poate fi marit). Se utilizează pentru aceasta următorul procedeu de etichetare:

a) Se marchează intrarea a cu $+$.

b) Un vîrf x fiind marcat, se va marca:

— cu $+$ x oricare vîrf y nemarcat cu proprietatea că arcul $u = (x, y)$ este nesaturat, adică $f(u) < c(u)$;

— cu $-$ x oricare vîrf y nemarcat cu proprietatea că arcul $u = (y, x)$ are un flux nenul, adică $f(u) > 0$.

Dacă prin acest procedeu de marcare se etichetează ieșirea b , atunci fluxul f_b obținut la pasul curent nu este maxim.

Se va considera un lanț format din vîrfuri etichetate (ale caror etichete au respectiv semnele $+$ sau $-$) care unește a cu b și care poate fi ușor găsit dacă se urmăresc etichetele vîrfurilor sale în sensul de la b către a .

Fie v acest lanț. Să notăm cu v^{+} mulțimea arcelor (x, y) ale lui v , unde marcarul lui y are semnul $+$, deci care sînt orientate în sensul de la a către b și cu v^{-} mulțimea arcelor (x, y) ale lui v , unde marcarul lui y are semnul $-$, deci care sînt orientate în sensul de la b către a .

Calculăm $\varepsilon_1 = \min_{u \in v^{+}} (c(u) - f(u))$, $\varepsilon_2 = \min_{u \in v^{-}} f(u)$ și $\varepsilon = \min(\varepsilon_1, \varepsilon_2)$.

Din modul de etichetare rezultă $\varepsilon > 0$. Vom mari cu ε fluxul pe fiecare arc $u \in v^{+}$ și îl vom micșora cu ε pe fiecare arc $u \in v^{-}$, obținînd la ieșire un flux egal cu $f_b + \varepsilon > f_b$. Se repeta aplicarea pasului 2 cu fluxul nou obținut.

Dacă însă prin aplicarea pasului 2 nu mai putem marca ieșirea b , am obținut un flux maxim și ne oprim. În plus, mulțimea arcelor care unesc vîrfurile marcate cu vîrfurile nemarcate constituie o tăietură de capacitate minimă.

Din modul de definire a lui ε obținem un nou flux f' care verifică $0 \leq f'(u) \leq c(u)$ pentru orice arc $u \in U$. În plus, condiția de conservare în fiecare vîrf $x \neq a, b$ este în continuare verificată. Într-adevăr, dacă lanțul v trece prin x și cele două arce incidente cu x din v aparțin amîndouă fie lui v^- , fie lui v^+ , se mărește cu ε (sau se micșorează cu ε) fluxul pe un arc de intrare și fluxul pe un arc de ieșire din x , deci condiția C este verificată pentru noul flux f' . Dacă cele două arce incidente cu x aparțin unul lui v^- și altul lui v^+ , atunci se micșorează, respectiv se mărește cu ε fluxul pe două arce, care sînt ambele sau arce de intrare în x sau arce de ieșire din x . Fluxul la ieșire crește deoarece $f'_b = f_b + \varepsilon > f_b$, ultimul arc al lanțului v care intră în b aparținînd mulțimii v^+ .

În cazul rețelei de transport cu fluxul reprezentat în figura III.10, aplicînd procedeul de etichetare expus obținem că ieșirea b poate fi etichetată pe lanțul $[(a, 2), (2, 6), (6, b)]$. Toate arcele acestui lanț au sensul de la a către b și etichetele vîrfurilor sînt trecute lângă vîrfurile respective în figura III.10.

Găsim $\varepsilon = \varepsilon_1 = \min(8 - 4, 4 - 2, 4 - 3) = 1$, deci vom mări fluxul cu cîte o unitate pe arcele $(a, 2)$, $(2, 6)$, $(6, b)$, obținînd fluxul din figura III.11

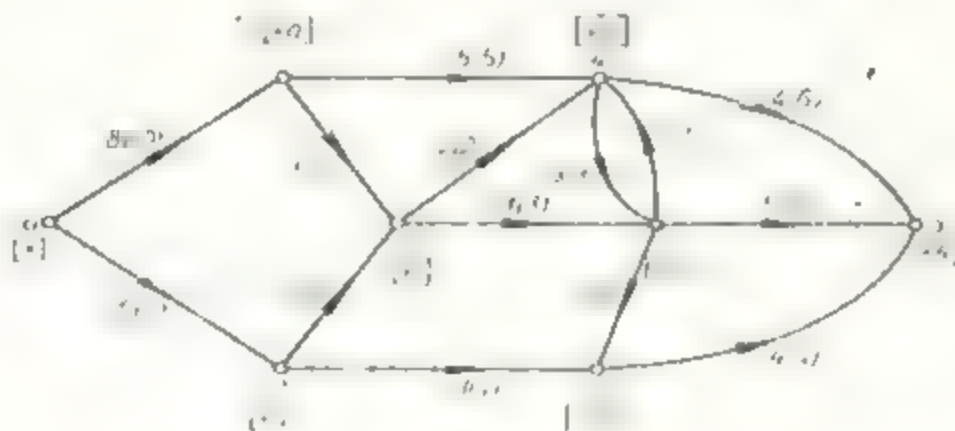


Fig. III.11

Ieșirea poate fi din nou etichetată, așa cum rezultă din figura III.11. Pentru a găsi lanțul nesaturat v de la a la b , procedăm astfel: b are marcajul $+4$, deci ultimul arc al lanțului v este $(4, b)$, vîrfurile 4 are eticheta $+5$, deci arcul anterior lui $(4, b)$ este $(5, 4)$ și nu $(4, 5)$ care are sensul de la b către a . Vîrfurile 5 are eticheta -6 , deci următorul arc este $(5, 6)$; vîrfurile 6 are marcajul $+2$, deci următorul arc este $(2, 6)$, vîrfurile 2 are marcajul $+a$ și deci ultimul arc este $(a, 2)$. Scriind arcele obținute în ordinea inversă găsim $v = [(a, 2), (2, 6), (6, 5), (5, 4), (4, b)]$. Calculăm $\varepsilon_1 = \min(8 - 5, 4 - 3, 1 - 0, 6 - 4) = 1$ și $\varepsilon_2 = \min(1) = 1$, deoarece $v^- = \{(5, 6)\}$.

Rezultă $\varepsilon = 1$, deci vom mări fluxul cu cîte o unitate pe arcele $(a, 2)$, $(2, 6)$, $(5, 4)$ și $(4, b)$ și îl vom micșora cu o unitate pe arcul $(5, 6)$, obținînd noul flux reprezentat în figura III.12.

Acum ieșirea nu mai poate fi etichetată, deci fluxul obținut este maxim.

Mulțimea vîrfurilor neetichetate este $A = \{4, 5, 6, b\}$, deci $\omega^-(A) = \{(1, 4), (3, 4), (3, 5), (2, 6)\}$, de capacitate $c(\omega^-(A)) = 14$. Arcurile acestei tăieturi sînt traversate de curba închisă desenată în figura III.12 în jurul ieșirii b .

Fluxul la ieșire este $f_b = 5 + 5 + 4 = 14 = c(\omega^-(A))$. Conform teoremei III.3.2 pentru orice flux și orice tăietură de suport A există inegalitatea: $f_b \leq c(\omega^-(A))$. Deoarece am găsit un flux și o tăietură pentru care această inegalitate să fie chiar egalitate, rezultă că fluxul obținut în figura III.12 este maxim, iar tăietura $\omega^-(A)$ are o capacitate minimă.

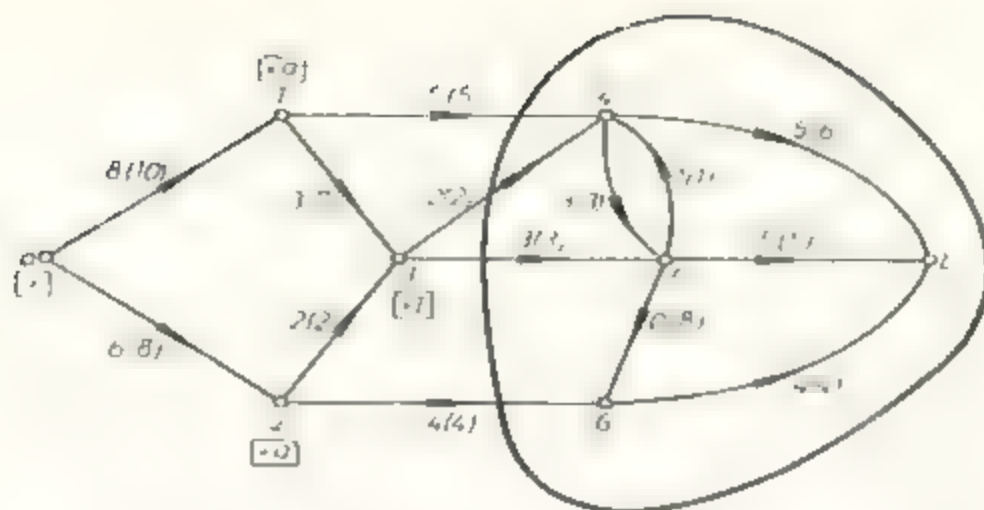


Fig. III.12

Teorema III.3.3. Algoritmul lui Ford-Fulkerson are un număr finit de pași pentru orice rețea de transport cu capacități numere întregi. În momentul cînd prin procedeul de etichetare de la pasul 2) nu mai putem eticheta ieșirea rețelei, fluxul obținut este maxim, iar mulțimea arcelor care unesc vîrfurile etichetate cu vîrfurile care nu au putut fi etichetate formează o tăietură de capacitate minimă. *Demonstrație* Vom arata mai întîi că algoritmul are un număr finit de pași, majorat de exemplu de $\sum_{u \in \omega^+(b)} c(u)$, adică de suma capacităților arcelor care intră în b .

Într-adevăr, conform teoremei III.3.2 există inegalitatea $f_b \leq c(\omega^-(A))$, deci $\max f_b \leq c(\omega^-(A))$, unde $\omega^-(A)$ este o tăietură oarecare a rețelei. Dacă alegem $A = \{b\}$, rezulta că $\omega^-(A) = \omega^-(b)$. Obținem $\max f_b \leq c(\omega^-(A)) = \sum_{u \in \omega^-(b)} c(u)$.

Inițial se pleacă cu fluxul nul pe fiecare arc sau cu un flux oarecare, deci $f_b \geq 0$ și la fiecare pas fluxul f_b crește cu ε . Deoarece capacitățile arcelor sînt numere întregi și nenegative, rezulta că valorile fluxului pe arce și valorile lui ε sînt numere întregi, iar $\varepsilon > 0$ implică ε întreg, $\varepsilon \geq 1$.

Deci la fiecare pas fluxul f_b crește cu cel puțin o unitate, deci numărul de pași ai algoritmului este majorat de capacitatea unei tăieturi.

Să arătăm că, în momentul în care ieșirea nu mai poate fi marcată, fluxul obținut este maxim, iar mulțimea arcelor care unesc vîrfurile marcate cu vîrfurile nemarcate formează o tăietură de capacitate minimă. Pentru aceasta, să notăm cu A mulțimea vîrfurilor care nu pot fi etichetate cu algoritmul descris. Obținem $a \notin A$ și $b \in A$, deoarece am presupus că ieșirea rețelei nu poate fi marcată. Deci $\omega^-(A)$ formează o tăietură a rețelei de transport. Pentru orice arc $u \in \omega^-(A)$ obținem $f(u) = c(u)$, deoarece din $u = (x, y)$ rezulta că $x \notin A$ și $y \in A$. Dacă $f(u) < c(u)$ ar rezulta că vîrfurile y ar putea fi etichetate, ceea ce contrazice faptul că $y \in A$. Pentru orice arc $u \in \omega^+(A)$ obținem $f(u) = 0$, deoarece dacă $u = (y, x)$ rezultă că $y \in A$ și $x \notin A$. Dacă $f(u) > 0$, vîrfurile y ar putea fi etichetate plecînd de la vîrfurile etichetate x , ceea ce contrazice faptul că $y \in A$. Deci putem scrie, conform egalității din (5):

$$f_b = \sum_{u \in \omega^-(A)} f(u) = \sum_{u \in \omega^-(A)} c(u) = c(\omega^-(A)).$$

Însă pentru orice flux f și orice tăietură $\omega^-(A)$ existînd inegalitatea $f_b \leq c(\omega^-(A))$, rezultă că fluxul f_b este maxim, iar tăietura $\omega^-(A)$ are o capacitate minimă.

Am obținut în același timp și următorul rezultat :

Corolar. Pentru orice rețea de transport, valoarea maximă a fluxului la ieșire este egală cu capacitatea minimă a unei tăieturi, adică :

$$\max f_v = \min c(\omega(A)).$$

Acest rezultat a fost dedus în cazul capacităților întregi, însă el este valabil pentru orice funcție de capacitate $c: U \rightarrow \mathbb{R}_+$.

Algoritmul lui Ford-Fulkerson permite găsirea fluxului maxim după un număr finit de pași în orice rețea de transport G pentru care capacitățile arcelor sînt numere raționale (vezi problema III.3.2).

Pentru a programa efectiv la calculator algoritmul de flux maxim trebuie să se utilizeze o structură de date adecvată structurii de rețea. De exemplu, arcele rețelei pot fi reprezentate prin extremitatea inițială, extremitatea finală, capacitatea și fluxul lor în această ordine sub forma unei liste liniare. O altă listă poate memora marcasele tuturor vîrfurilor diferite de intrarea rețelei :

De exemplu, pentru rețeaua din figura III.11 această reprezentare poate arăta astfel

$$L_1: \underbrace{[a \mid 1 \mid 10 \mid 8]}_{(a, 1)} \underbrace{[a \mid 2 \mid 8 \mid 5]}_{(a, 2)} \underbrace{[1 \mid 3 \mid 7 \mid 3]}_{(1, 3)}$$

unde primele patru poziții ale listei L_1 memorează informația relativă la arcul $(a, 1)$ în ordinea menționată, următoarele 4 cea relativă la arcul $(a, 2)$ următoarele 4 cea relativă la arcul $(1, 3)$ ș.a.m.d.

Pentru a ușura găsirea arcelor care pleacă din fiecare vîrf se poate folosi o altă listă L_2 , care indică pe care poziție a listei L_1 încep arcele care pleacă din fiecare vîrf diferit de ieșire.

Pentru exemplul considerat, vom avea $L_2: \underbrace{[1 \mid 9 \mid 17]}_{a \mid 1 \mid 2}$ deoarece reprezentarea

arcelor care pleacă din a începe în poziția 1 a listei L_1 , reprezentarea arcelor care pleacă din 1 începe în poziția 9 a listei L_1 , a arcelor care pleacă din 2 în poziția 17 ș.a.m.d.

Folosirea listei L_2 asociată cu L_1 face de fapt inutilă reprezentarea în lista L_1 a extremităților inițiale : $a, 1, 2, \dots$, ale arcelor rețelei G . Pentru a ușura găsirea arcelor care intră într-un vîrf x , ceea ce este necesar pentru procesul de marcare, se poate utiliza o listă L'_1 și o listă L'_2 asociată care să memoreze arcele care intră în vîrfurile 1, vîrfurile 2, \dots , în b , în această ordine.

În fine, lista pentru marcare în cazul rețelei din figura III.11 arată astfel

$$L_2: [+a \mid +a \mid +1 \mid +5 \mid -6 \mid +2 \mid +4]$$

Poziția întâi din L_2 este rezervată pentru marcasele vîrfurilor 1, poziția a II-a, pentru marcasele vîrfurilor 2, \dots , ultima poziție pentru marcasele ieșirii b . Plecînd de la lista L_2 , se poate găsi simplu lanțul nesaturat care permite creșterea fluxului la ieșire : Ultimul marcare din L_2 este $+4$, deci ultimul arc este $(4, b)$. Pe poziția a IV-a în L_2 găsim $+5$, deci arcul anterior este $(5, 4)$. Pe poziția 5 găsim în L_2 numărul -6 . Acest număr fiind negativ, arcul anterior nu este $(6, 5)$, ci este $(5, 6)$ ș.a.m.d.

În cazul rețelei din figura III.12 lista L , arată astfel :

$$L, \begin{bmatrix} +a & +a & +1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

deoarece vîrfurile 4, 5, 6 și b nu au putut fi marcate.

Aplicații

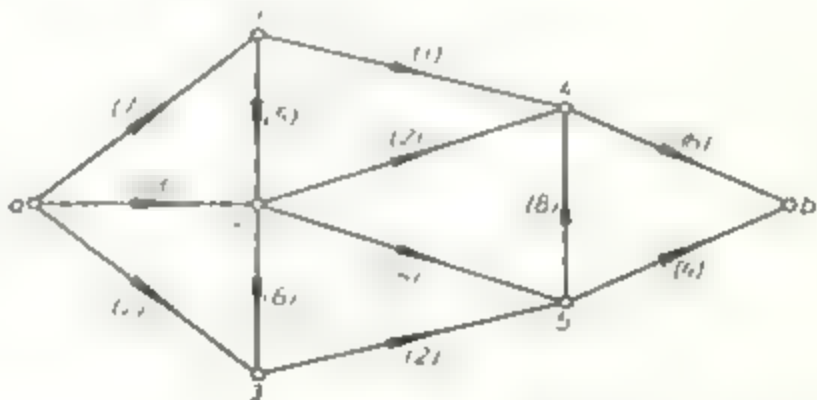
Unele probleme de transport sînt probleme de flux. De exemplu sa presupunem că există n depozite conținînd respectiv cantitățile a_1, \dots, a_n dintr-un anumit produs, care trebuie transportate la m destinații, care pot primi cel mult b_1, \dots, b_m unități din produs. Problema maximizării cantității totale care poate fi transportată se reduce la problema găsirii unui flux maxim în rețeaua de transport definită astfel: Graful G are vîrfurile $x_1, \dots, x_n, y_1, \dots, y_m$, o intrare a și o ieșire b . Mulțimea U este formată din: arcele (a, x_i) cu o capacitate egală cu a_i pentru $1 \leq i \leq n$; arcele (y_j, b) cu o capacitate egală cu b_j pentru $1 \leq j \leq m$ și toate arcele de forma (x_i, y_j) pentru $1 \leq i \leq n$ și $1 \leq j \leq m$, care au o capacitate corespunzătoare capacității de transport de la x_i la y_j (problema III.3.3).

Vom putea satisface toate cererile în y_1, \dots, y_m dacă oferta este mai mare sau egală cu cererea, ceea ce se scrie $\sum_{i=1}^n a_i \geq \sum_{j=1}^m b_j$ și dacă toate capacitățile de transport sînt suficient de mari. Determinarea taieturii de capacitate minimă prezintă interes într-o serie de aplicații (vezi problemele III.3.5 și III.3.6).

Probleme

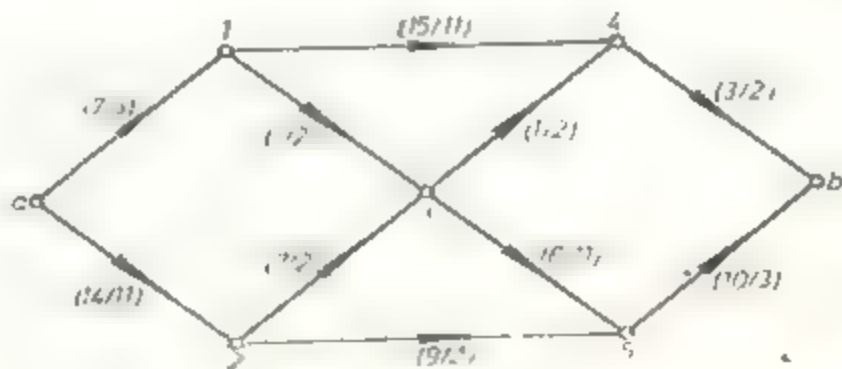
III.3.1. Sa se găsească fluxul maxim în rețeaua de transport din figura III.13.

Fig. III.13



III.3.2. Aceeași problemă pentru rețeaua de transport cu capacități numere fracționare din figura III.14.

Fig. III.14



Deoarece atât domeniul de definiție al unei funcții booleene, cât și domeniul de valori sînt mulțimi finite, există un număr finit de funcții booleene cu un domeniu de definiție și un domeniu de valori date. Astfel, mulțimea funcțiilor definite pe B_2^n cu valori în B_2 conține 2^{2^n} funcții. Aceste funcții pot fi numerotate, asociind la fiecare funcție un număr între 0 și $2^{2^n} - 1$. O funcție corespunde unui șir de 2^n valori 0 sau 1; aceste valori formează un număr binar cu 2^n cifre. Acest număr va fi denumit indexul funcției respective.

În tabelul 1.7 sînt date toate funcțiile de o variabilă în număr de $2^{2^1} = 4$. Cele patru funcții pot fi interpretate astfel:

- f_0 este funcția constantă 0,
- f_1 este funcția identică,
- f_2 este funcția complement,
- f_3 este funcția constantă 1.

x	f_0	f_1	f_2	f_3
0	0	0	1	1
1	0	1	0	1

Tabelul 1.7

În tabelul 1.8 sînt prezentate toate funcțiile de două variabile. Asupra celor 16 funcții de două variabile se pot face mai multe observații. Astfel, funcțiile f_0 și f_{16} sînt funcțiile constante 0 și respectiv 1. Funcțiile f_1 și f_5 sînt funcțiile identice cu x_1 și respectiv x_2 , iar f_{15} și f_{14} sînt funcțiile identice cu complementul variabilei x_1 și respectiv x_2 . Funcțiile f_6 și f_{12} nu depind de nici o variabilă, iar funcțiile f_2 , f_{10} și f_{13} nu depind decît de una din variabile. Cele alte funcții depind de ambele variabile.

x_1x_2	00	01	10	11
f_0	0	0	0	0
f_1	0	0	0	1
f_2	0	0	1	0
f_3	0	0	1	1
f_4	0	1	0	0
f_5	0	1	0	1
f_6	0	1	1	0
f_7	0	1	1	1
f_8	1	0	0	0
f_9	1	0	0	1
f_{10}	1	0	1	0
f_{11}	1	0	1	1
f_{12}	1	1	0	0
f_{13}	1	1	0	1
f_{14}	1	1	1	0
f_{15}	1	1	1	1

Tabelul 1.8

O altă formă de reprezentare a funcțiilor booleene utilizează expresiile booleene. O expresie booleană formată cu variabilele x_1, x_2, \dots, x_n generează o funcție booleană de n variabile booleene, care asociază fiecărui sistem de n elemente 0 sau 1 valoarea expresiei booleene pentru acest sistem de valori.

III 3.3. Să se găsească planul de transport pentru a transporta o cantitate maximă de produse din x_1, x_2, x_3 în y_1, y_2, y_3 , știind că se pot transporta din x_i în y_j următoarele cantități maxime, date în tabelul:

	y_1	y_2	y_3
x_1	12	21	46
x_2	15	18	34
x_3	9	26	12

La intersecția liniei x_i cu coloana y_j se găsește cantitatea de produse care pot fi transportate din x_i în y_j .

Disponibilul la centrele x_i este dat de $a_1 = 60, a_2 = 24$ și $a_3 = 36$, iar cererile în centrele y_j sînt următoarele $b_1 = 33, b_2 = 19, b_3 = 68$. Să se transforme această problemă într-o problemă de flux maxim într-o rețea de transport și să se determine un plan optim de transport utilizînd algoritmul lui Ford-Fulkerson.

III 3.4. Fie G o rețea de transport cu intrarea a , ieșirea b și $\omega(A)$ o tăietură oarecare cu $a \notin A$ și $b \in A$. Să se arate că orice drum $D = (a, \dots, b)$ de la a la b în rețea conține cel puțin două vîrfuri vecine x_i și x_{i+1} , astfel încît arcul $(x_i, x_{i+1}) \in \omega(A)$.

III 3.5. Două localități a și b sînt legate printr-o rețea de drumuri cu 6 puncte de intersecție. Pentru a ajunge din a în b trebuie să urmăm drumurile de la a la b din graful din figura III.15. Costurile de instalare a unor posturi de control al circulației pe arcele acestui graf sînt indicate prin numerele scrise în paranteze lîngă arcele respective.

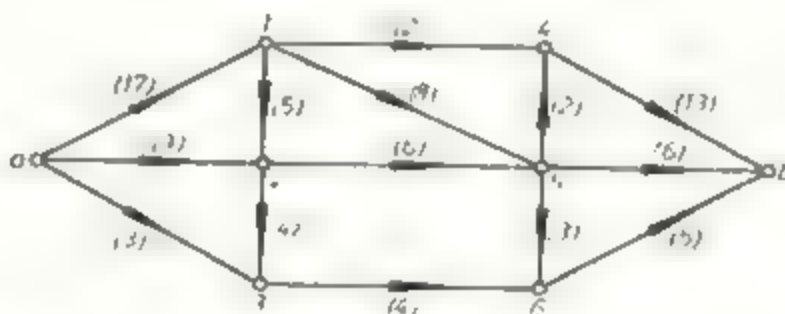


Fig. III.15

Să se determine pe ce arce ale rețelei trebuie instalate posturile de control, astfel încît să poată fi supravegheate toate drumurile posibile de deplasare din a în b , iar costul total de instalare să fie minim.

III 3.6. Se consideră graful de activități din figura III 16, unde vîrfurile 1 și 7 reprezintă începerea lucrării, vîrfurile 2 și 6 reprezintă terminarea ei, iar timpii operatori sînt numerele scrise în dreptul fiecărui arc.

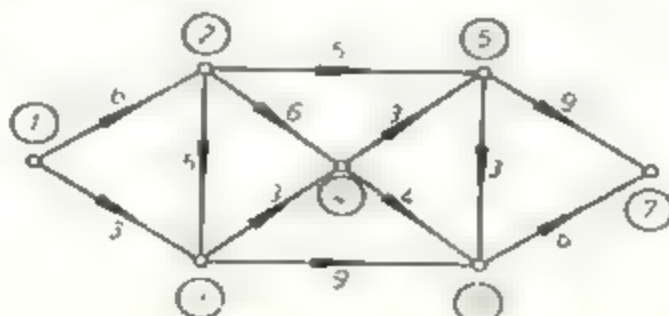


Fig. III 16

Costul necesar pentru a determina accelerarea operațiilor programului cu câte o unitate de timp este trecut în tabelul următor :

Operația	Costul	Operația	Costul
(1,2)	6	(3,6)	3
(1,3)	2	(4,5)	4
(2,3)	6	(4,6)	1
(2,4)	1	(5,6)	1
(2,5)	3	(5,7)	1
(3,4)	5	(6,7)	5

Să se determine care operații trebuie să fie accelerate cu câte o unitate de timp, astfel încât durata de execuție a întregii lucrări să se scurteze cu o unitate de timp și costul total pentru scurtarea duratelor acestor operații să fie minim.

IV. ALGORITMI ȘI METODE DE REPREZENTARE

IV.1. Noțiunea de algoritm

IV.1.1. Definiția algoritmului

Noțiunea de algoritm este o noțiune matematică foarte veche. Cuvântul „algoritm” este de origine arabă. El derivă din numele matematicianului „Abu Ja'far Mohammed ibn Mûsâ al Horezmî” care a scris o carte celebră intitulată „Kitab al jabr w'al-muqabala”. Din titlul acestei cărți provine cuvântul algebră.

În evul mediu se folosea termenul de „algorism” cu înțelesul de proces al efectuării operațiilor aritmetice cu ajutorul cifrelor arabe. Se presupune că din asocierea cuvântului algorism cu domeniul lui de referință, aritmetica, a rezultat termenul algoritm. Începînd cu anul 1950 în toate manualele de specialitate cuvântul algoritm este frecvent asociat cu procesul de aflare a celui mai mare divizor comun a două numere naturale, așa numitul „algoritm al lui Euclid”. De asemenea, regulile operațiilor aritmetice sînt denumite algoritmi de efectuare a operațiilor respective.

Noțiunea de algoritm nu are o definiție matematică. În aceeași situație se află și alte noțiuni din matematică, cum ar fi noțiunea de mulțime.

Prin *algoritm* se accepta să se înțeleagă un sistem de calcule, care, pentru o anumită clasă de probleme, din condițiile inițiale ale problemei permite să se obțină soluția problemei respective, cu ajutorul unui șir finit și ordonat de operații univoc determinate, efectuate mecanic, fără aportul creator al omului.

Există totuși definiții matematice riguroase pentru unele categorii de algoritmi: funcții recursive, mașini Turing, algoritmi normali ai lui A. A. Markov. S-a demonstrat că aceste clase de algoritmi sînt echivalente. S-a emis ipoteza că oricare dintre aceste clase definește noțiunea de algoritm. Pîna în prezent această ipoteză nu a fost infirmată, deoarece nu s-a găsit nici un algoritm care să nu poată fi reprezentat conform regulilor acestor clase.

Un algoritm este compus din unul sau mai mulți pași, un pas reprezentînd efectuarea unei singure operații din șirul celor care alcătuiesc algoritmul.

Exemple

1. Algoritmul împărțirii întregi a două numere naturale

Se știe că împărțirea întregă a două numere constă din efectuarea unor scăderi succesive, pîna cînd descăzutul devine mai mic decît scăzătorul. Pentru fiecare scădere care se efectuează, descăzutul este rezultatul scăderii precedente, iar scăzătorul este împărțitorul. Rezultatul ultimei scăderi efectuate este tocmai restul împărțirii celor două numere, iar numărul de scăderi efectuate reprezintă citul împărțirii.

Pașii acestui algoritm sînt constituiți de operațiile de scădere și de operațiile de comparare a descăzutului cu scăzătorul. Este evident că șirul acestor ope-

rații este finit, deoarece descăzutul se micșorează cu fiecare nouă scădere, în timp ce scăzătorul rămâne neschimbat.

Fie, de exemplu, numerele 17 și 7. Pașii algoritmului care duc la aflarea cîtului și restului împărțirii sînt prezentați în tabelul IV.1.

Operația	Pasul	Numarul scăderii
scădere	$17 - 7 = 10$	1
comparare	$10 < 7$ nu	—
scădere	$10 - 7 = \boxed{3}$	2
comparare	$3 < 7$ da	—
(descăzutul este mai mic decît scazatorul)		

Tabelul IV.1

Numarul de scăderi efectuate este 2, iar rezultatul ultimei scăderi efectuate este 3, deci cîtul împărțirii numărului 17 prin 7 este 2, iar restul este 3

2. Algoritmul lui Euclid

Acest algoritm se folosește pentru obținerea celui mai mare divizor comun a două numere naturale.

Notînd cele două numere naturale prin m și n , vom presupune că m este mai mare decît n .

Algoritmul consta din efectuarea unui șir de împărțiri întregi pina cînd se obține un rest nul. Pentru fiecare împărțire care se efectuează, împărțitorul este restul împărțirii precedente, iar deîmpărțitul este împărțitorul din împărțirea precedentă. Împărțitorul din ultima împărțire efectuată constituie cel mai mare divizor comun al celor două numere.

Pașii acestui algoritm sînt constituiți de operațiunile de împărțire și de verificare a anularii restului. Deoarece restul unei împărțiri este mai mic decît împărțitorul, șirul de resturi al împărțirilor succesive este strict descrescător, astfel că numarul de împărțiri din algoritm este finit.

Fie, de exemplu, numerele 78 și 30. Pașii algoritmului care conduc la aflarea celui mai mare divizor comun al acestor numere sînt prezentați în tabelul IV.2

Pasul	Operația
$78 : 30 = 2$ rest 18	împărțire
$18 \neq 0$ nu	verificare
$30 : 18 = 1$ rest 12	împărțire
$12 \neq 0$ nu	verificare
$18 : 12 = 1$ rest 6	împărțire
$6 \neq 0$ nu	verificare
$12 : \boxed{6} = 2$ rest 0	împărțire
$0 = 0$ da	verificare

Tabelul IV.2

IV.1.2 Proprietățile algoritmilor

Orice algoritm trebuie să se bucure de următoarele trei proprietăți fundamentale, numite proprietăți de baza ale algoritmilor.

1. Claritatea Orice algoritm trebuie să fie caracterizat printr-o descriere precisă, riguroasă, fără ambiguități a tuturor acțiunilor care urmează să se execute. Cu alte cuvinte, un algoritm, datorită caracterului sau de automatism, trebuie să precizeze în mod univoc toate etapele de calcul pe care le va urma executantul algoritmului (omul sau mașina). De aceea, această proprietate este denumită uneori și unicitate.

2. Generalitatea. Un algoritm este util dacă rezolvă nu numai o problemă particulară, concretă, ci o întreaga clasă de probleme asemănătoare. Aceasta înseamnă că un algoritm trebuie să se aplice la o mulțime de sisteme de date inițiale. Aceasta mulțime poartă numele de domeniu de aplicabilitate al algoritmului.

De exemplu, algoritmul lui Euclid se poate aplica la orice pereche de numere naturale. Vom spune deci ca domeniul de aplicabilitate al algoritmului lui Euclid este mulțimea perechilor de numere naturale. Această proprietate este cunoscută și sub numele de universalitate.

3. Eficacitatea Orice algoritm urmărește prin execuția sa obținerea unui anumit rezultat. Pentru aceasta nu este suficient ca acțiunile algoritmului să fie bine determinate, ci trebuie ca pentru orice sistem de date inițiale numărul de acțiuni (pași) care urmează să se execute să fie finit. De aceea, această proprietate poartă denumirea de finitudine. La exemplele de algoritmi prezentate anterior s-a aratat ca numărul de pași corespunzători unui sistem oarecare de date inițiale este finit.

IV.1.3. Structura algoritmilor

Acțiunile componente ale unui algoritm se efectuează asupra unor date inițiale sau asupra unor rezultate intermediare ale operațiilor anterioare. Atât datele cât și rezultatele intermediare apar ca valori ale unor variabile. O variabilă are, în cadrul unui algoritm, o semnificație deosebită de aceea din matematică. Astfel, în timp ce în matematica o variabilă reprezintă o nedeterminată cu care se pot face operații matematice fara a fi cunoscuta valoarea sa, într-un algoritm variabilele sînt utilizate pentru a denumi date sau rezultate intermediare. Deci, o variabilă este destinată să aibă o anumită valoare. Această valoare poate fi totuși schimbată pe parcursul algoritmului. Este cazul așa-numitelor variabile de lucru. O variabilă de lucru este folosită pentru reținerea unui rezultat intermediar și poate fi refolosită pentru reținerea altui rezultat intermediar, atunci cînd rezultatul anterior nu mai este necesar pentru alte calcule.

Acțiunile unui algoritm se realizează sub forma unor operații. Aceste operații constituie pașii algoritmului.

Operațiile care pot apărea într-un algoritm sînt de două categorii : operații de calcul și operații de decizie

O operație de calcul constă în efectuarea calculelor indicate de o expresie simbolică, înlocuind fiecare variabilă cu valoarea sa. Rezultatul acestor calcule adică valoarea expresiei respective, este reținut sub forma valorii unei variabile. Se spune, de obicei, ca valoarea expresiei este atribuită variabilei respective.

Notăția utilizată pentru operațiile de calcul poate fi dedusă din următoarele exemple :

$$x \leftarrow y^2 + 2$$

$$\alpha \leftarrow \sqrt{\beta^2 - 1/2}$$

$$S \leftarrow LUNGIME \times LĂȚIME$$

$$X_0 \leftarrow \frac{x_1 + x_2}{2}$$

Observații

1. Variabilele pot fi notate atât prin litere, cât și prin cuvinte. Se pot utiliza și indici.
2. Simbolul „ \leftarrow ”, care se citește „la valoarea”, este întrebuințat pentru a marca atribuirea unei valori variabilei indicate. În matematică pentru același scop se utilizează simbolul „ $=$ ”. Simbolul „ $=$ ” este însă folosit și pentru relația de egalitate. Pentru a evita ambiguitatea în cadrul unui algoritm, se preferă folosirea a doua simboluri distincte, astfel că simbolul „ $=$ ” va fi folosit numai pentru a marca relația de egalitate.

Într-o operație de calcul se poate atribui o nouă valoare unei variabile a cărei valoare este utilizată în cadrul calculului respectiv.

De exemplu, operația de calcul

$$X \leftarrow X^2$$

atribuie variabilei X o nouă valoare, egală cu vechea valoare ridicată la pătrat. În acest fel iese în evidență modul de succesiune a valorilor unei variabile. Astfel, dacă înainte de efectuarea operației de atribuire prezentate, variabila X are valoarea 3, după efectuarea operației, aceeași variabilă va avea valoarea 9.

Un rol deosebit de important în structura unui algoritm îl au operațiile de decizie.

În general, prin operație de decizie se înțelege determinarea valorii logice de adevăr a unei propoziții. Propozițiile analizate de operațiile de decizie sînt propoziții enunțative, care nu pot fi decît adevărate sau false. De obicei, aceste propoziții enunță că un obiect are o anumită proprietate (de exemplu : valoarea variabilei X este pozitivă, variabila Y are ca valoare un număr întreg). De cele mai multe ori, propozițiile asupra cărora se aplică operația de decizie se referă la o relație între două obiecte. De exemplu : valoarea variabilei X este egală cu valoarea variabilei Y , valoarea variabilei X se divide prin valoarea variabilei Y etc.

Rezultatul unei operații de decizie îl constituie valoarea „adevărat” sau „fals” a propoziției analizate. În cadrul acestei operații se calculează valorile diferitelor expresii care constituie obiectele relațiilor respective, ținînd cont de valorile variabilelor care apar în aceste expresii. Astfel, propoziția

$$X + 2 > Y \quad 1$$

are valoarea logică „adevărat” dacă, de exemplu, variabila X are valoarea 7, iar variabila Y valoarea 2 și valoarea logică „fals” dacă variabilele X și Y au valorile 3 și, respectiv, 8.

O importanță deosebită în descrierea unui algoritm o are și specificarea succesiunii de efectuare a operațiilor componente.

Înlanțuirea pașilor unui algoritm poate fi indicată implicit sau explicit.

Astfel, succesiunea implicită de efectuare a pașilor unui algoritm este dată de ordinea de prezentare a acestor pași în cadrul algoritmului.

Specificarea explicită a succesiunii de efectuare a unor pași apare în cazul operațiilor de decizie. În acest caz se specifică (explicit) care pas urmează să fie executat dacă rezultatul operației de decizie este „adevărat” și care pas dacă rezultatul operației este „fals”. Deoarece acești doi pași următori sînt obligatoriu diferiți, operațiile de decizie reprezintă ramificații în succesiunea de pași ai unui algoritm. Să exemplificăm cele două tipuri de operații în cadrul algoritmului împărțirii întregi (cu rest).

Să notăm variabilele acestui algoritm astfel.

D — deîmpărțitul;
 I — împărțitorul;
 C — cîțul;
 R — restul.

Presupunind ca variabilele D și I au deja valorile corespunzătoare datelor algoritmului, pașii acestuia sînt :

Pasul 1. $C \leftarrow 0$;

Pasul 2. $R \leftarrow D$;

Pasul 3. $R < I$, dacă este adevărat urmează pasul 7,
 dacă este fals urmează pasul 4;

Pasul 4. $R \leftarrow R - I$;

Pasul 5. $C \leftarrow C + 1$;

Pasul 6. $R < I$, dacă este adevărat urmează pasul 7,
 dacă este fals urmează pasul 4;

Pasul 7. Terminarea algoritmului.

Să observăm că pașii 1, 2, 4 și 5 specifică operațiile de calcul, în timp ce pașii 3 și 6 reprezintă operații de decizie. În tabelul IV.3 se prezintă succesiunea pașilor algoritmului în cazul numerelor 17 și 7.

Pasul	Valorile variabilelor înainte de execuția pasului				Rezultatul operației de decizie
	D	I	C	R	
1	<u>17</u>	<u>7</u>	—	—	—
2	17	7	0	—	—
3	17	7	0	17	fals
4	17	7	0	17	—
5	17	7	0	10	—
6	17	7	1	10	fals
4	17	7	1	10	—
5	17	7	1	3	—
6	17	7	2	3	adevărat
7	17	7	<u>2</u>	<u>3</u>	—

Tabelul IV.3

Este posibil ca să se specifice explicit următorul pas și în cazul operațiilor de calcul. Astfel, în exemplul anterior, observînd identitatea pașilor 3 și 6, se poate elimina pasul 6, specificînd la pasul 5 că urmează pasul 3.

Rezultă astfel următorul algoritm :

Pasul 1. $C \leftarrow 0$;

Pasul 2. $R \leftarrow D$;

Pasul 3. $R < I$, dacă este adevărat urmează pasul 6,
dacă este fals urmează pasul 4;

Pasul 4. $R \leftarrow R - I$;

Pasul 5. $C \leftarrow C + 1$, urmează pasul 3;

Pasul 6 Terminarea algoritmului.

De asemenea, este posibilă specificarea implicită a pasului următor și în cazul operațiilor de decizie. Totuși, deoarece într-un pas corespunzător unei operații de decizie se specifică doi pași următori, rămâne necesară specificarea explicită a unuia dintre acești doi pași. Astfel, în exemplul prezentat poate fi modificat pasul 3 în modul următor:

Pasul 3. dacă $R < I$ urmează pasul 6.

IV.1.4. Clasificarea algoritmilor

În funcție de structura lor algoritmii pot fi împărțiți în mai multe clase.

Algoritmi liniari sînt acei algoritmi care sînt alcătuiți numai din operații de calcul. Absența operațiilor de decizie din cadrul algoritmilor liniari are ca efect execuția pașilor acestor algoritmi într-o singură succesiune. În această categorie intra, de exemplu, algoritmul pentru calculul valorii unei expresii, cum ar fi algoritmul pentru calculul valorii polinomului $ax^2 + bx + c$, prezentat în continuare.

Pasul 1. $v \leftarrow a$;

Pasul 2. $v \leftarrow v \times x + b$;

Pasul 3. $v \leftarrow v \times x + c$;

Pasul 4. Terminarea algoritmului.

Algoritmi liniari sînt cei mai simpli algoritmi

Algoritmi cu ramificații reprezintă acei algoritmi care cuprind și operații de decizie printre operațiile de calcul. În acest caz, în funcție de valorile variabilelor și de rezultatele operațiilor de decizie, pentru un algoritm cu ramificații există mai multe posibilități în ceea ce privește ordinea de execuție a pașilor săi.

Algoritmi aciclici cuprind acea categorie de algoritmi cu ramificații, pentru care în cadrul execuției nu se poate efectua de mai multe ori un același pas. Algoritmul rezolvării unei ecuații de gradul doi, $ax^2 + bx + c = 0$, este un exemplu de astfel de algoritm.

Pasul 1. $\text{DELTA} \leftarrow b^2 - 4ac$;

Pasul 2. dacă $\text{DELTA} < 0$ urmează pasul 5;

Pasul 3. $x_1 \leftarrow \frac{-b + \sqrt{\text{DELTA}}}{2a}$;

Pasul 4. $x_2 \leftarrow \frac{-b - \sqrt{\text{DELTA}}}{2a}$; urmează pasul 7;

Pasul 5. $\text{RE} \leftarrow \frac{-b}{2a}$;

Pasul 6. $\text{IM} \leftarrow \frac{\sqrt{-\text{DELTA}}}{2a}$;

Pasul 7. Terminarea algoritmului.

Se observă că nu sînt posibile decît două succesiuni de pași și anume: 1, 2, 3, 4, 7 în cazul rădăcinilor reale și 1, 2, 5, 6, 7 în cazul rădăcinilor complexe.

Algoritmii pentru care există posibilitatea repetării execuției unuia sau mai multor pași, alcătuiesc categoria *algoritmilor ciclici*. Succesiunea de pași care poate fi executată în mod repetat poartă denumirea de *ciclu*. Numărul de repetări ale unui ciclu poate fi fix sau variabil.

Cea mai mare parte a algoritmilor utilizați în practică este reprezentată de algoritmii ciclici. Acești algoritmi permit o descriere concisă a unor succesiuni cu numeroși pași. Algoritmii ciclici sînt indicați în special pentru a fi executați cu ajutorul calculatoarelor electronice. În acest caz numărul mare de pași de efectuat este compensat de viteza mare de execuție a operațiilor.

Exemplele de algoritmi prezentate la început, algoritmul lui Euclid și algoritmul împărțirii întregi constituie algoritmi ciclici. Ciclul algoritmului lui Euclid este reprezentat de pașii corespunzători efectuării unei împărțiri, iar ciclul algoritmului împărțirii întregi este format din pașii corespunzători efectuării unei scăderi.

Un algoritm ciclic poate avea mai multe cicluri, acestea putînd fi incluse unul în altul.

IV.1.5. Exerciții






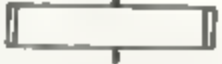
1. Fie N un număr natural. Este corect următorul algoritm :
 Pasul 1. $M \leftarrow 2N + 1$
 Pasul 2. Dacă $M = 0$ atunci urmează pasul 4 ;
 Pasul 3. $M \leftarrow M - 2$, urmează pasul 2 ;
 Pasul 4. Terminarea algoritmului.
2. Se dă algoritmul :
 Pasul 1. $I \leftarrow 1$;
 Pasul 2. $S \leftarrow 1$;
 Pasul 3. $S \leftarrow S \times I$;
 Pasul 4. $I \leftarrow I + 2$;
 Pasul 5. Dacă $I \leq 5$ atunci urmează pasul 3 ;
 Pasul 6. Terminarea algoritmului
 Care este valoarea lui S la terminarea algoritmului ?
3. Să se scrie pașii algoritmului lui Euclid pentru numerele 459 și 170.
4. Se dă polinomul $P(x) = a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$. Să se alcătuiască un algoritm care să calculeze valoarea $P(x_0)$.
5. Să se găsească un algoritm care să determine cel mai mare număr dintre trei numere date.
6. Folosind schema lui Horner să se conceapă un algoritm care să calculeze cîțul și restul împărțirii polinomului $ax^3 + bx^2 + cx + d$ la $x - x_0$.
7. Să se scrie un algoritm pentru calculul sumei primilor n termeni ai unei progresii geometrice cu rația q și primul termen a .
8. Să se conceapă un algoritm pentru calculul numărului de combinații C_n^k .

IV.2. Metode de reprezentare a algoritmilor

IV.2.1. Scheme logice

Cea mai simplă metodă de reprezentare a algoritmilor este metoda folosită în exemplele anterioare, care utilizează pentru descrierea fiecărui pas al algoritmului un limbaj apropiat de cel obișnuit.

Schemele logice constituie forme grafice mai sugestive de reprezentare a algoritmilor, cu o răspîndire largă, o schema logică fiind alcătuită din blocuri între care se stabilesc legături orientate (săgeți). Blocurile au diferite forme grafice, în funcție de acțiunile (pași) pe care le (ii) reprezintă (tabelul IV 4)

Denumirea blocului	Simbolul
Terminal	
De calcul	
De decizie	
De intrare	
De ieșire	
De procedură	

Tabelul IV.4

Operațiile de calcul sînt reprezentate prin blocuri de calcul, iar operațiile de decizie prin blocuri de decizie.

Pentru a pune în evidență începutul sau sfîrșitul unei scheme logice se utilizează blocuri terminale.

Datele de intrare ale algoritmului precum și rezultatele acestuia sînt puse în evidență cu ajutorul unor blocuri de intrare-ieșire.

Legaturile care se stabilesc între blocuri reprezintă succesiunea efectuării acțiunilor reprezentate de blocurile respective

Se remarcă, la blocurile de decizie, legăturile corespunzătoare celor două rezultate posibile ale operației de decizie.

Schema logica a algoritmului împărțirii întregi este prezentată în figura IV.1.

Pentru a simplifica trasarea legaturilor, precum și pentru a ușura urmărirea succesiunii pașilor unui algoritm se obișnuiește ca aceste linii de legătură să fie parcurse de sus în jos sau de la stînga la dreapta. Liniile de legătură care necesită să fie parcurse în alt mod trebuie să fie desenate sub forma de săgeți. Este însă recomandabil ca toate liniile să fie trasate sub forma de săgeți.

Pentru a nu diminua claritatea unei scheme logice se recomandă evitarea intersectării liniilor de legătură. Dacă totuși unele intersectări nu pot fi eliminate, există un simbol special, denumit conector, care permite întreruperea unei linii de legătură. Un conector este alcătuit dintr-un cerculeț în care se înscrie o literă sau cifră. Se presupune că între două simboluri conector care conțin

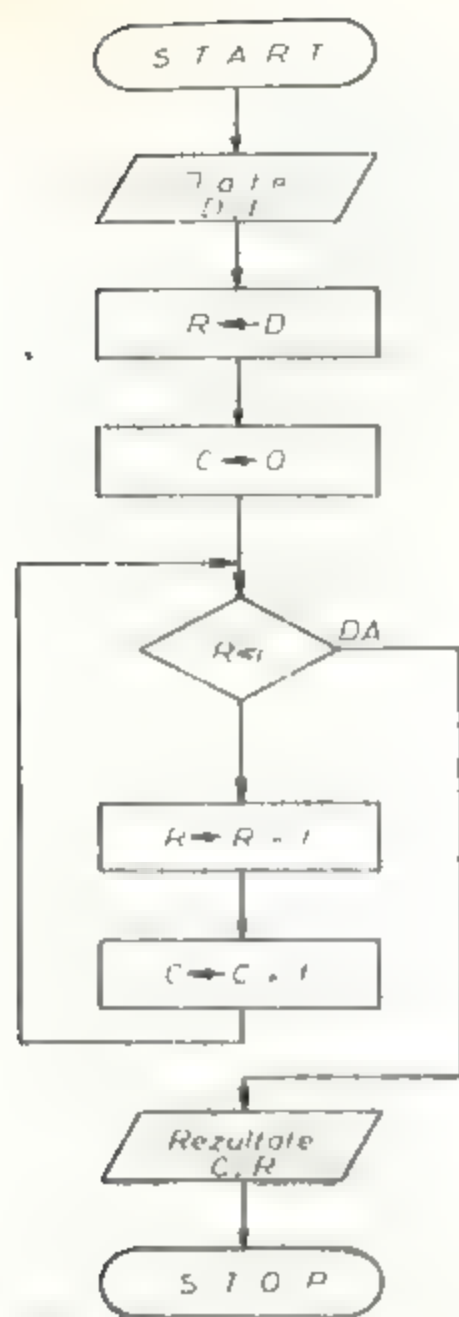


Fig. IV.1

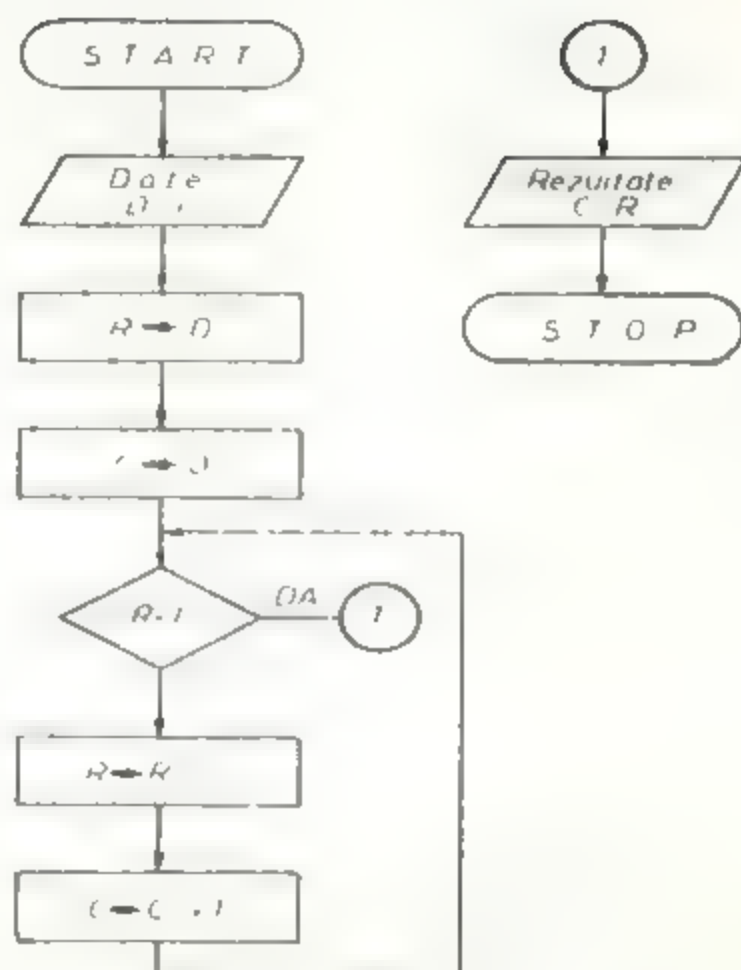


Fig. IV.2

aceeași literă sau aceeași cifră există o linie de legătură netrasată. În figura IV 2 se prezintă un exemplu de utilizare a acestui tip de simboluri.

Pe măsură ce un algoritm este mai complex, schema sa logică cuprinde mai multe blocuri, iar legăturile necesare sînt mai numeroase. Pentru a simplifica descrierea unui algoritm mai complex, se pot folosi operații complexe, a căror efectuare să reprezinte de fapt execuția unui întreg algoritm. De exemplu, în cadrul algoritmului lui Euclid se poate folosi operația de împărțire cu rest, pentru a cărei efectuare se poate utiliza algoritmul împărțirii întregi prezentat anterior. Pentru reprezentarea acestor operații speciale se folosește în schemele logice un tip de bloc denumit bloc de procedură (vezi tabelul IV.4). În figura IV.3 se poate urmări reprezentarea algoritmului lui Euclid în care s-a utilizat algoritmul împărțirii cu rest. Se observă că blocul de procedură din figura IV 3 descrie aplicarea algoritmului împărțirii întregi pentru valorile variabilelor M și N , obținînd ca rezultate valorile variabilelor P și Q .

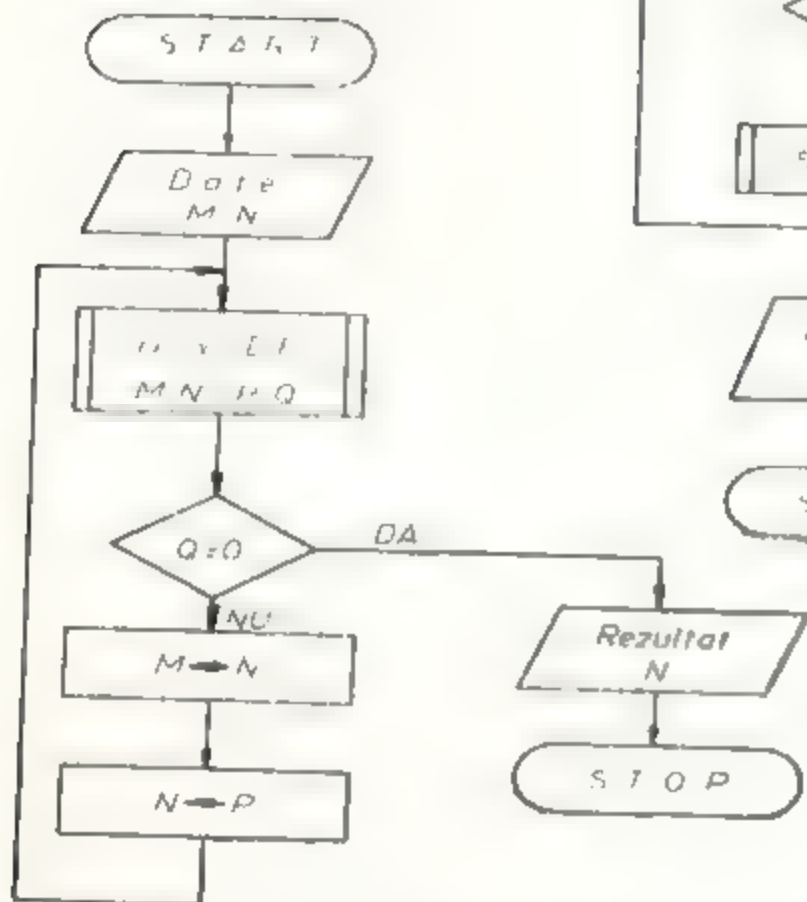


Fig. IV.3

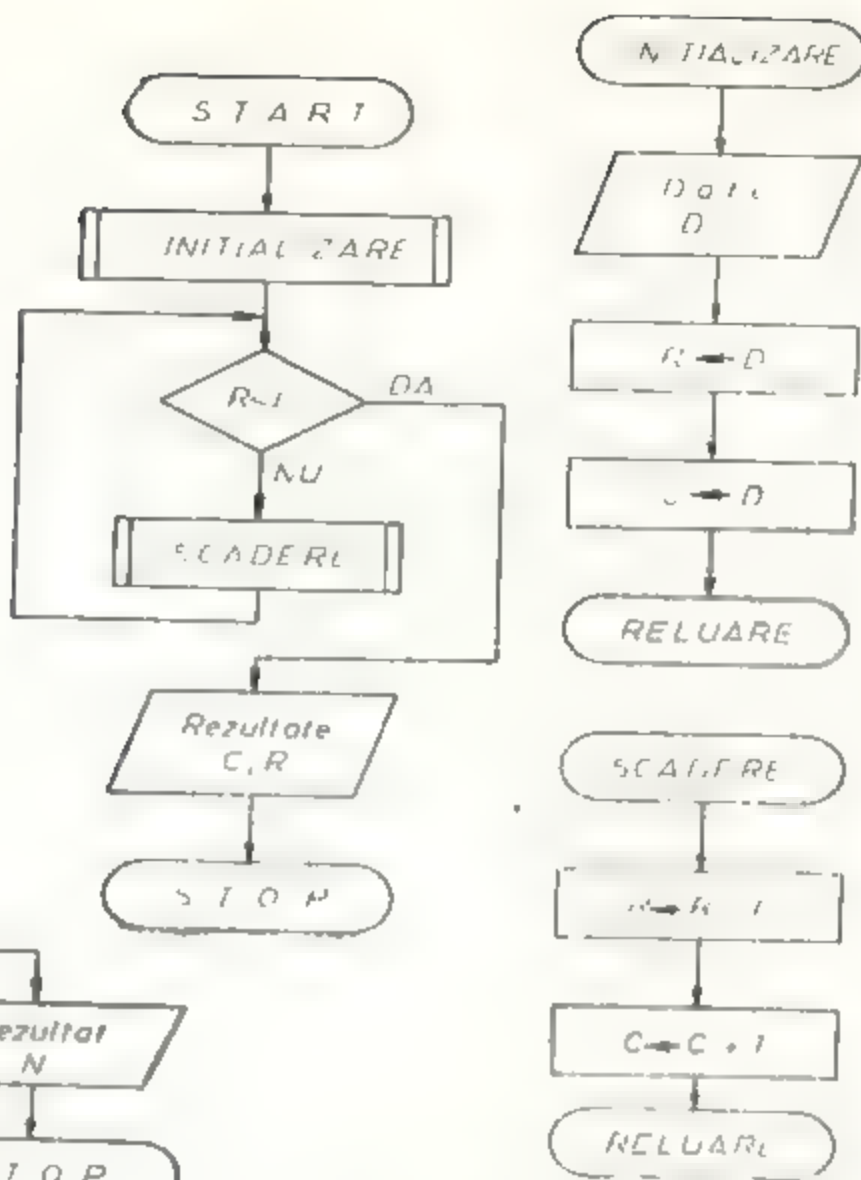


Fig. IV.4

Un bloc de procedură poate fi însă folosit și pentru a nota o porțiune dintr-o schemă logică, porțiune care este descrisă separat.

Cu o notație de acest fel algoritmul împărțirii întregi apare ca în figura IV.4.

Realizarea unui algoritm corect și a unei scheme logice clare într-un timp scurt și cu un efort cât mai mic constituie scopul cercetărilor în domeniul programării. S-au obținut bune rezultate folosind metoda programării structurate, apărută în 1965.

Conform metodei programării structurate un algoritm poate să fie realizat folosind o combinație de trei tipuri de structuri elementare: liniară, alternativă și repetitivă (fig. IV.5). S-a arătat că orice algoritm descris de o schemă logică poate fi transformat astfel ca să fie reprezentat printr-o schemă logică structurată. În figura IV.6 se prezintă algoritmul împărțirii întregi în forma structurată. Singura modificare necesară (față de fig. IV.1) a fost la blocul de decizie, unde relația respectivă s-a înlocuit cu relația complementară.

Structura alternativă corespunde în limbajul curent unei propoziții condiționale de forma: dacă C atunci B altfel A .

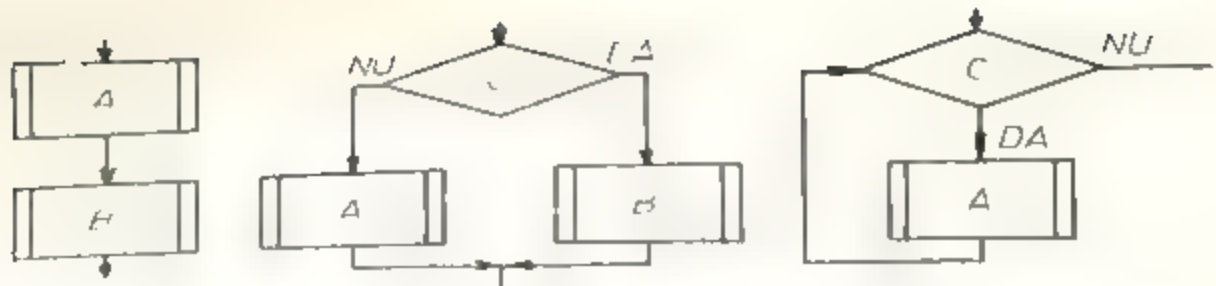


Fig IV 5

În cazul algoritmilor, structura alternativă corespunde unei operații de decizie de forma: dacă propoziția C este adevărată, atunci urmează pasul i ; altfel urmează pasul j .

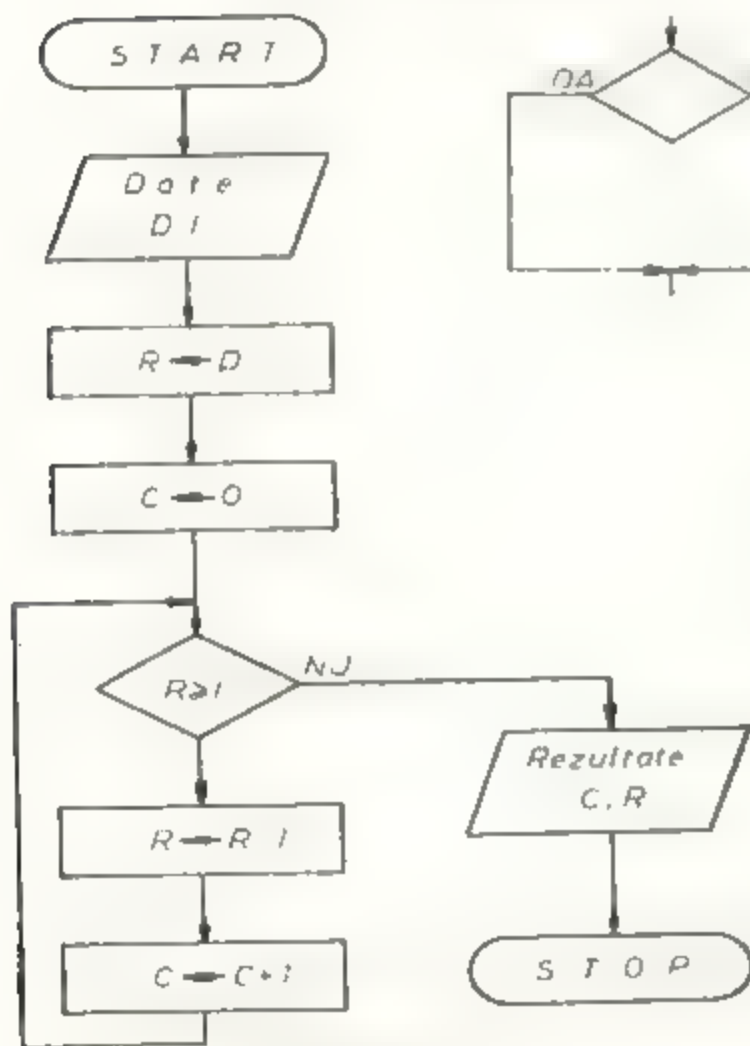


Fig IV.6

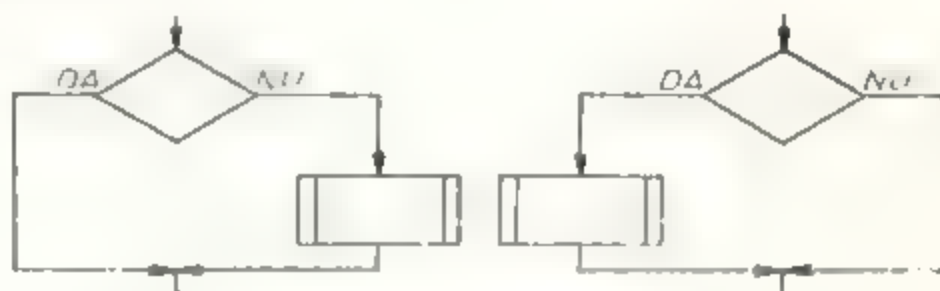


Fig IV 7

Pentru ca structura alternativă să fie corect construită este însă necesar ca să urmeze același pas, atât după succesiunea de pași corespunzătoare blocului de procedură A , cât și după succesiunea corespunzătoare blocului de procedură B .

Structura alternativă poate fi utilizată și sub o formă simplificată, dacă una dintre alternative nu conține nici un pas. Grafic, această situație se prezintă ca în figura IV.7.

Structura respectivă corespunde în limbajul curent unei propoziții de forma: atât timp cât este adevărată C execută A .

În cazul algoritmilor structura repetitivă corespunde ciclurilor de pași, propoziția C determinând numărul de repetări ale ciclului.

IV 2.2. Exerciții

1. Să se alcătuiască schema logică a unui algoritm care să calculeze $n!$ când se dă n .
2. Să se alcătuiască schema logică a algoritmului pentru rezolvarea ecuației $ax^2 + bx + c = 0$.
3. Ce valoare are S când se termină execuția algoritmilor reprezentați de schemele logice din figura IV.8.

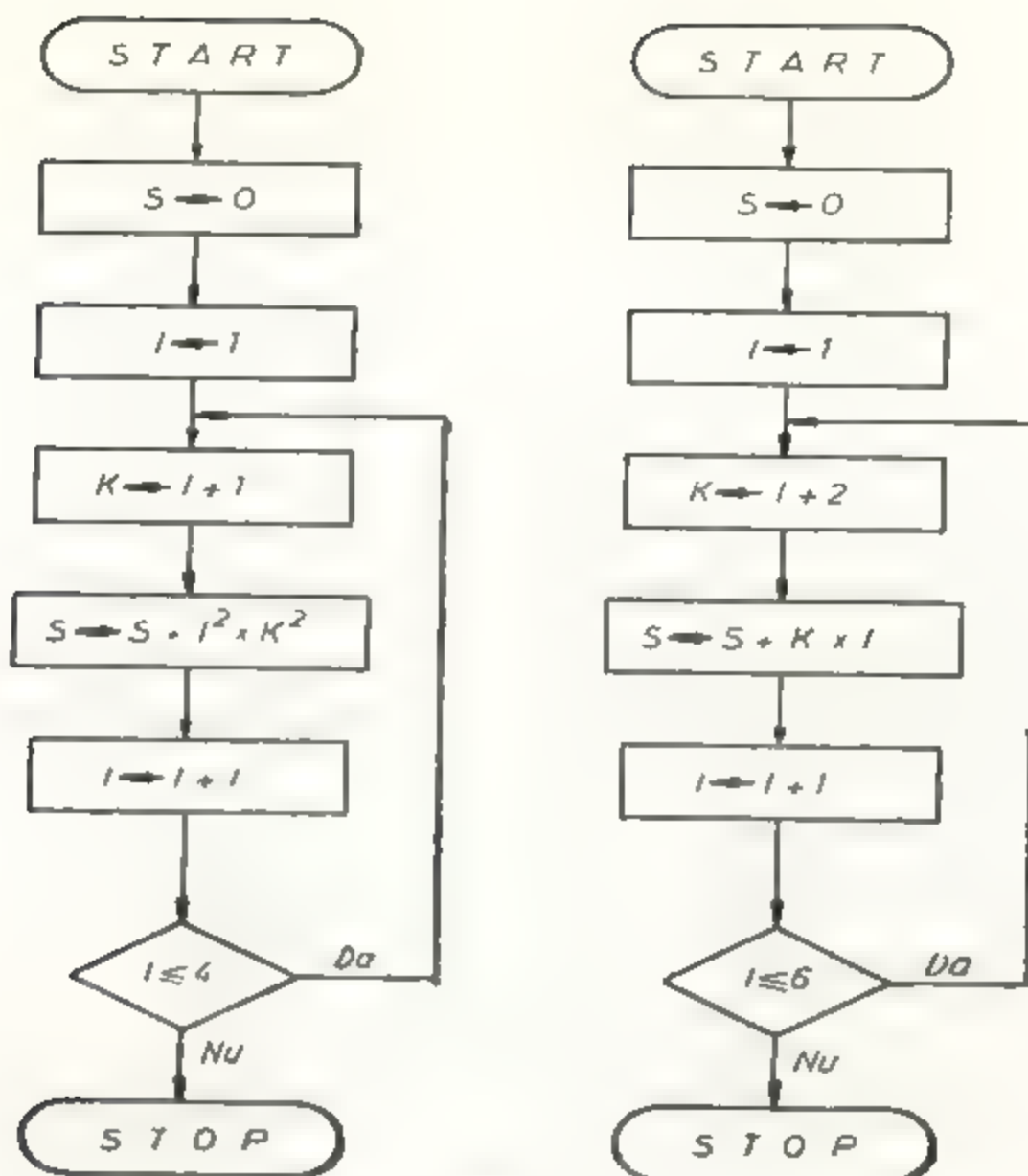


Fig. IV.8

4. Să se alcătuiască schemele logice ale algoritmilor descriși în exercițiile 4, 5, 6 și 7 din paragraful anterior.

5. Se dă șirul $\{a_k\}$, $k = 1, 2, \dots$ al cărui termen sînt definiți de relația :

$$a_{k+1} = \frac{1}{2} \left(a_k + \frac{b}{a_k} \right); \quad a_1 = 1.$$

Să se alcătuiască algoritmul pentru calculul valorii termenului a_n .

Să se deseneze schema logică a acestui algoritm.

6. Să se alcătuiască algoritmi și schemele logice pentru calculul următoarelor sume :

a) $S = 1^2 + 3^2 + 5^2 + \dots + (2n + 1)^2$

b) $S = 1! + 2! + 3! + \dots + n!$

c) $S = C_n^0 + C_n^1 + C_n^2 + \dots + C_n^n$.

Indicații și răspunsuri

Cap. I

- 1.1.3. 1. a) $x \sqcup (y \sqcap x) = x \sqcup (x \sqcap y) = x$ și $(x \sqcup y) \sqcap x = x \sqcap (x \sqcup y) = x$, b) $p \sqcup (y \sqcap \sqcap z) = y \sqcap z$ și $(p \sqcup y) \sqcap z = y \sqcap z$; c) $x \sqcup (y \sqcap u) = (x \sqcup y) \sqcap (x \sqcup u)$ și $(x \sqcup y) \sqcap u = x \sqcup y$
2. $x \sqsubseteq z \Leftrightarrow x \sqcap z = x$ astfel că $(x \sqcup y) \sqcap z = (x \sqcap z) \sqcup (y \sqcap z) = x \sqcup (y \sqcap z)$. 3. b) $n = 2$, $a_1 = p$ și $a_n = u$. Fie $a_2 \neq a_1$ și $a_n \neq a_{n-1}$. $a_1 \sqcup a_2 = u = a_n \Rightarrow \bar{a}_1 = a_n$, iar $a_2 \sqcup \bar{a}_1 = p = a_1 \Rightarrow \bar{a}_1 = a_1$, imposibil, deoarece $a_1 \neq a_n$. 4. b) 8; c) 5. 5. a) $aa = a(a + 0) = a$; b) $\bar{a}\bar{a} = 0$ și $\bar{a} + \bar{a} = 1 \Rightarrow \bar{a} = a$, c) $a0 = 0 + 0a = 0$; d) $a + 1 = 1 + 1 \cdot a = 1$; e) $1 \cdot 0 = 0$ și $1 + 0 = 1 \Rightarrow 1 \neq 0$; f) $1 = 0 = 0$. 6. \emptyset , planul. 8. $a \cdot n = n \Rightarrow n \subseteq a$ pentru orice a și $a + u = u \Rightarrow a \subseteq u$ pentru orice a . 9. a) 1; b) 0; c) 1.
- 1.2.4. 1. a) x ; b) b ; c) 0; d) x . 4. a) 1; b) 0; c) 0. 6. a) da, b) da, c) nu
- 1.3.3. 1. a) x ; b) $x_1x_2x_3x_4$; c) $\bar{a}\bar{b} + \bar{c}\bar{d}$; d) $\bar{a}\bar{c} + \bar{b}\bar{c} + \bar{a}\bar{b}\bar{c}$. 2. a) x ; b) $x_1x_2x_3x_4$; c) $(\bar{a} + \bar{c})(\bar{b} + \bar{c})(\bar{a} + \bar{d})(\bar{b} + \bar{d})$; d) $(a + b)(b + c)(a + c)$. 3. b, c, e, f. 4. a) $abc + a\bar{b}\bar{c} + \bar{a}\bar{b}c + \bar{a}b\bar{c}$; b) $xyz + \bar{x}yz + x\bar{y}\bar{z}$; c) $x_1x_2x_3 + x_1\bar{x}_2x_3 + \bar{x}_1\bar{x}_2x_3$. 5. a) $(a + b + \bar{c})(a + \bar{b} + c)(a + b + c)$; b) $(x + y + z)(\bar{x} + \bar{y} + \bar{z})(\bar{x} + y + \bar{z})(x + y + \bar{z})(\bar{x} + \bar{y} + z)$; c) $(x_1 + x_2 + x_3)(\bar{x}_1 + \bar{x}_2 + \bar{x}_3)(\bar{x}_1 + \bar{x}_2 + x_3)(\bar{x}_1 + \bar{x}_2 + \bar{x}_3)$. 6. a) $\bar{a}\bar{b}\bar{c}\bar{d} + \bar{a}\bar{b}c\bar{d} + \bar{a}b\bar{c}\bar{d} + \bar{a}b\bar{c}d + \bar{a}b\bar{c}d + \bar{a}b\bar{c}d + \bar{a}b\bar{c}d + \bar{a}b\bar{c}d$; b) $(a + b + c + d)(a + b + \bar{c} + \bar{d})(a + \bar{b} + c + d)(a + \bar{b} + \bar{c} + \bar{d})(\bar{a} + b + c + d)(\bar{a} + b + \bar{c} + \bar{d})(\bar{a} + \bar{b} + c + d)(\bar{a} + \bar{b} + \bar{c} + \bar{d})$
- 1.4.3. 1. a) 0; b) $x_1x_2x_3$; c) $x + y + \bar{z}$; d) $\bar{a} + \bar{b} + \bar{c}$. 2. a) $\bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}\bar{d}$; b) xyz ; c) $x_1x_2 + \bar{x}_1 + \bar{x}_2$; d) $\bar{a}\bar{b} + \bar{a}\bar{c}\bar{d}$; e) \bar{x} . 3. a) $\bar{x}\bar{y} + \bar{x}\bar{z}$; b) $\bar{a}\bar{b} + \bar{a}\bar{c} + \bar{a}\bar{c}$; c) $x_1x_2 + \bar{x}_1x_3$. 4. a) 1; b) $x_1x_2x_3$; c) $\bar{a}\bar{b}$. 5. a) $ac + \bar{b}\bar{c}$; b) $\bar{b}\bar{d}$; c) $x + y + z$. 6. b) $\bar{x}yz + x\bar{y}\bar{z} + x\bar{y}\bar{z} + x\bar{y}\bar{z}$; $(x + y + z)(x + y + \bar{z})(x + \bar{y} + z)(\bar{x} + y + z)$; c) $xy + xz + yz$. 7. a) $f_1 = x_1x_2x_3 + x_1x_2x_3 + x_1x_2x_3 + x_1x_2x_3 + x_1x_2x_3 + x_1x_2x_3 + x_1x_2x_3 + x_1x_2x_3$, $f_2 = x_1x_2x_3 + x_1x_2x_3 + x_1x_2x_3 + x_1x_2x_3 + x_1x_2x_3 + x_1x_2x_3 + x_1x_2x_3 + x_1x_2x_3$, $f_3 = x_1x_2x_3 + x_1x_2x_3 + x_1x_2x_3 + x_1x_2x_3 + x_1x_2x_3 + x_1x_2x_3 + x_1x_2x_3 + x_1x_2x_3$; b) $f_1 = (x_1 + x_2 + \bar{x}_3)(x_1 + \bar{x}_2 + x_3)(\bar{x}_1 + \bar{x}_2 + x_3)$, $f_2 = (x_1 + \bar{x}_2 + x_3)(\bar{x}_1 + x_2 + x_3)$, $f_3 = (x_1 + x_2 + x_3)(x_1 + \bar{x}_2 + x_3)(x_1 + \bar{x}_2 + \bar{x}_3)(\bar{x}_1 + \bar{x}_2 + \bar{x}_3)(\bar{x}_1 + \bar{x}_2 + \bar{x}_3)$; c) $f_1 = x_1x_2 + x_1x_3 + \bar{x}_2x_3$, $f_2 = x_1 + \bar{x}_1x_2 + x_1x_3$, $f_3 = x_1x_3 + \bar{x}_1x_2x_3$. 8. a) $A\bar{B}\bar{C} + A\bar{B}\bar{C} + A\bar{B}C + A\bar{B}C = AC + \bar{A}B$; b) $A\bar{B}\bar{C} + A\bar{B}C + A\bar{B}C = A\bar{B}C + \bar{B}C$; c) $A\bar{B}\bar{C} + A\bar{B}C + A\bar{B}C$. 9. a) C ; b) $A\bar{B} + \bar{A}B$; c) $\bar{A}\bar{B}$; d) $AC + AB + \bar{B}C$. 11. a) $\bar{a}\bar{b}\bar{c}\bar{d} + \bar{a}\bar{b}\bar{c}d + \bar{a}\bar{b}c\bar{d} + \bar{a}\bar{b}cd + \bar{a}b\bar{c}\bar{d} + \bar{a}b\bar{c}d + \bar{a}b\bar{c}d + \bar{a}b\bar{c}d$, b) $\bar{a}\bar{b}\bar{c}\bar{d} + \bar{a}\bar{b}\bar{c}d + \bar{a}\bar{b}c\bar{d} + \bar{a}\bar{b}cd + \bar{a}b\bar{c}\bar{d} + \bar{a}b\bar{c}d + \bar{a}b\bar{c}d + \bar{a}b\bar{c}d$, c) $\bar{a}\bar{b}\bar{c}\bar{d}e + \bar{a}\bar{b}\bar{c}\bar{d}\bar{e} + \bar{a}\bar{b}\bar{c}d\bar{e} + \bar{a}\bar{b}\bar{c}de + \bar{a}\bar{b}c\bar{d}\bar{e} + \bar{a}\bar{b}c\bar{d}e + \bar{a}\bar{b}cde + \bar{a}\bar{b}cde + \bar{a}b\bar{c}\bar{d}\bar{e} + \bar{a}b\bar{c}\bar{d}e + \bar{a}b\bar{c}d\bar{e} + \bar{a}b\bar{c}de$. 12. a) $c + d + \bar{a}\bar{b}$; b) $\bar{x}\bar{y} + xyz$; c) $\bar{x}z + xw$; d) x_1x_2 ; e) $\bar{b}\bar{d} + \bar{a}\bar{b}\bar{c}$.

1.5.6. 1. a) $x_1x_2 + x_2x_3 + \bar{x}_1\bar{x}_2 = x_2 + x_3$; b) $(x_1\bar{x}_2 + x_2x_3)[\bar{x}_1(\bar{x}_2x_3 + x_2)x_3 + x_2\bar{x}_3 + \bar{x}_2x_3] = \bar{x}_1\bar{x}_2(x_1 + x_3)$; c) $(\bar{x}_1 + x_2)(x_1x_3 + \bar{x}_1\bar{x}_2)(x_2\bar{x}_3 + \bar{x}_2x_3) = \bar{x}_1x_2\bar{x}_3$. 2. a) $\overline{a_1a_2 \cdot a_3 + a_3a_4 + a_4 + a_4 \cdot a_5}$; b) $\bar{b}c + a(b + c)(a + b + c) + abc$; c) $[(\bar{x}_3 + \bar{x}_2)x_4 + x_2x_3\bar{x}_4]x_1\bar{x}_5$. 3. a) $(\bar{a}b) \cdot (bc) = a + b\bar{c}$; b) $\bar{a}\bar{b}cd + \bar{a}b\bar{c}d + abcd + ab\bar{c}d = cd$; c) $abd + abd + a\bar{b}c + ac = ab + c$. 5. (vezi fig. V.1). 6. (vezi fig. V.2). 7. (vezi fig. V.3).

a	0	0	0	0	1	1	1	1
b	0	0	1	1	0	0	1	1
c	0	1	0	1	0	1	0	1
f	0	1	1	0	1	0	0	1

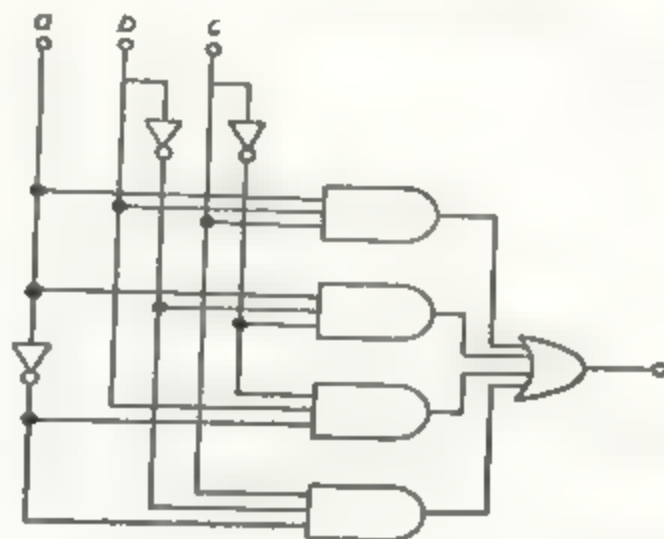


Fig. V.1

a	0	0	0	0	1	1	1	1
b	0	0	1	1	0	0	1	1
c	0	1	0	1	0	1	0	1
f	0	0	0	1	0	1	1	1

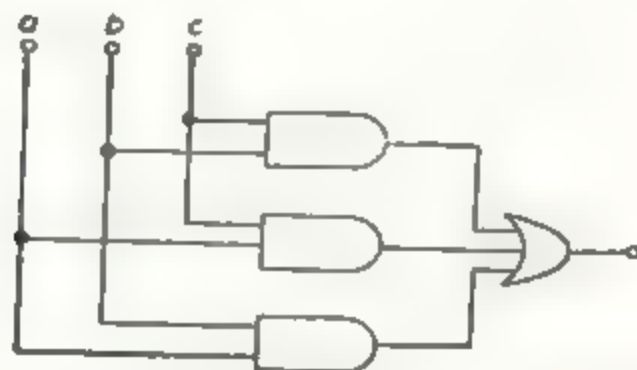


Fig. V.2

C_1C_2	D_1D_2	00	01	11	10
00	0	0	0	0	0
01	0	1	1	0	0
11	1	1	1	1	1
10	0	0	1	1	1

$$f = D_1D_2 + C_1D_1 + C_2D_2$$



Fig. V.3

II.1.1. Se obțin 13 lanțuri elementare, 7 care încep cu muchia $[1, 2]$ și 6 care încep cu muchia $[1, 3]$.

II.1.2. Se va raționa prin reducere la absurd.

II.1.3. Dacă G conține un vîrf izolat x , toate muchiile lui G unesc vîrfuri diferite de x , deci $m \leq C_{n-1}^2$.

II.1.4. Numărul maxim de muchii ale lui G se obține cînd fiecare componenta conexă este un subgraf complet. Presupunînd că există două componente care conțin n_1 , respectiv n_2 vîrfuri, astfel încît $n_1 \geq n_2 \geq 2$, vom face următoarea transformare: Suprimăm un vîrf x din componenta cu n_2 vîrfuri și îl ducem în componenta cu n_1 vîrfuri, unindu-l prin muchii cu toate cele n_1 vîrfuri deja existente. Se obține un nou graf G_1 cu m_1 muchii și

$$m_1 = m - (n_2 - 1) + n_1 = m + (n_1 - n_2) + 1 \geq m + 1,$$

ceea ce contrazice maximalitatea grafului G . Deci G are p componente care conțin respectiv $n - p + 1, 1, \dots, 1$ vîrfuri, cînd $m = C_{n-p+1}^2$.

II.1.5. Presupunînd prin reducere la absurd că G conține n vîrfuri de grade diferite, rezultă că șirul gradelor lui G este: $0, 1, 2, \dots, n-1$. Deci G conține în același timp un vîrf izolat și un vîrf adiacent cu toate celelalte vîrfuri, ceea ce este contradictoriu.

II.1.6. Pentru grafurile din figura II 14 bijecția f se definește de exemplu prin: $f(1) = a, f(2) = c, f(3) = e, f(4) = b, f(5) = d$ și pentru grafurile din figura II 15 prin: $f(1) = a, f(2) = b, f(3) = c, f(4) = d, f(5) = e, f(6) = f, f(7) = h, f(8) = i, f(9) = g, f(10) = j$, urmărind elefurile elementare din cele două grafuri.

II.1.7. Să presupunem că G nu este conex. Dacă vîrfurile x și y din \bar{G} nu sînt adiacente, rezultă că ele sînt adiacente în G .

Deci x și y aparțin unei aceleiași componente conexe a lui G . Graful G mai conține cel puțin o altă componentă conexă, deci există un vîrf z care nu aparține în graful G unei aceleiași componente cu x și cu y .

Rezultă că x și z , respectiv y și z nu sînt adiacente în G , adică $[x, z] \in \bar{U}$ și $[z, y] \in \bar{U}$, deci x și y sînt legate printr-un lanț de lungime egală cu doi. Deci \bar{G} este conex.

II.1.8. Fiecare din cele $C_n^2 = \frac{n(n-1)}{2}$ muchii ale grafului complet K_n poate fi aleasă sau nu

ca muchie a unui graf cu aceeași mulțime de vîrfuri.

Deci există $2^{C_n^2}$ grafuri cu n vîrfuri date.

II.1.9. Din definiția dată rezultă că relația binară este simetrică și reflexivă. Dacă $L_1 = [x, \dots, y]$ și $L_2 = [y, \dots, z]$ rezultă că $L = [x, \dots, y, \dots, z]$ este un lanț de la x la z , deci relația binară \sim este tranzitivă.

II.1.10. Să presupunem, prin reducere la absurd, că orice vîrf x al lui G are gradul $d(x) \geq 3$.

Fie:

$$L = [x_1, \dots, x_p]$$

unde $p \geq 4$, un cel mai lung lanț elementar care pleacă din x_1 . Rezultă că x_p este adiacent numai cu vîrfuri din mulțimea $\{x_1, \dots, x_{p-1}\}$. Vîrfurile x_p are gradul cel puțin 3 și este adiacent cu x_{p-1} , deci există doi indici $1 \leq r < s \leq p-2$ astfel încît x_p să fie adiacent cu x_r și x_s . Ciclurile elementare $C_1 = [x_s, x_{s+1}, \dots, x_p, x_r]$ și $C_2 = [x_r, x_{r+1}, \dots, x_p, x_s]$ sînt impare conform ipotezei. Însă ciclul elementar $C_3 = [x_r, x_{r+1}, \dots, x_{p-1}, x_p, x_s]$ este par, ceea ce contrazice ipoteza.

II.1.11. Vîrfurile 4, 5 și 6 nu pot aparține două cîte două unui aceluiași subgraf complet.

Fiecare din ele aparține unui număr de 6 subgrafuri complete cu 3 vîrfuri. Se obțin în total 18 subgrafuri complete cu 3 vîrfuri.

II.2.1. Cele două grafuri izomorfe din figura II 15 nu sînt hamiltoniene.

Graful din figura II 21 este hamiltonian, un ciclu hamiltonian fiind:

$$[1, 2, 12, 11, 10, 9, 3, 4, 5, 6, 7, 8, 1].$$

II.2.2. Din cele 8 cicluri hamiltoniene ale grafului dat există două cu un cost minim, egal cu 12 unități și anume: $[A, E, B, C, D, A]$ și $[A, D, C, B, E, A]$. Răspunde cerințelor problemei ciclul $[A, E, B, C, D, A]$.

II.2.3. Vîrfurile 2 și 6 avînd gradul egal cu 2, rezultă că ciclul hamiltonian conține muchiile $[P, 2]$, $[2, 4]$ și $[5, 6]$, $[6, 7]$.

Se obțin deci patru cicluri hamiltoniene, dintre care ciclul $[P, 2, 4, 5, 6, 7, 8, 11, 9, 10, 1, 3, P]$ are o lungime totală minimă, egală cu 33 unități.

II.2.4. Notînd cu $H(n)$ numărul ciclurilor hamiltoniene ale grafului K_n , obținem $H(3) = 1$ și $H(n+1) = nH(n)$, deoarece vîrfurile x_{n+1} poate fi intercalat în n noduri distincte printre vîrfurile x_1, \dots, x_n ale unui ciclu hamiltonian cu n vîrfuri pentru a obține un ciclu hamiltonian cu $n+1$ vîrfuri. Demonstrația rezultă prin inducție după n .

II.2.5. Dacă notăm cu X mulțimea vîrfurilor lui K_n , fie C un ciclu elementar cu k vîrfuri ($3 \leq k \leq n$), care formează mulțimea A , al lui K_n . Considerînd subgraful lui K_n indus de A , el este un graf complet care admite pe C drept ciclu hamiltonian. Deoarece A poate fi aleasă în X în C_n^k moduri, ținînd seama de problema precedentă, rezultă că numărul ciclurilor elementare cu k vîrfuri ale lui K_n este egal cu

$$C_n^k \frac{(k-1)!}{2} = \frac{1}{2} \cdot \frac{n(n-1) \dots (n-k+1)}{k}.$$

Însumînd aceste numere pentru $k = 3, \dots, n$ se obține rezultatul căutat.

II.3.1. $[1, 2, 3, 4, 5, 6, 4, 7, 6, 8, 7, 2, 10, 8, 9, 10, 1]$. Acest ciclu nu este unic.

II.3.2. Graful conține 6 vîrfuri de grad impar și anume: 2, 3, 4, 7, 8, 9, deci sînt necesare cel puțin 3 noi muchii pentru ca gradele tuturor vîrfurilor să fie pare, deci ca graful obținut să fie eulerian.

Dacă adăugăm de exemplu muchiile: $[2, 9]$, $[3, 7]$ și $[4, 8]$, această condiție va fi verificată și graful obținut are un ciclu eulerian.

II.3.3. Se vor adăuga trei noi muchii, de exemplu $[1, 2]$, $[6, 10]$ și $[3, 7]$.

II.3.4. Dacă G are gradele tuturor vîrfurilor numere pare, rezultă din teorema demonstrată că fiecare componentă conexă diferită de un vîrf izolat conține un ciclu eulerian. Dar fiecare ciclu

eulerian se poate scrie ca o reuniune de cicluri elementare, parcurgînd ciclul eulerian plecînd dintr-un vîrf x . Pentru a demonstra necesitatea, sã observãm cã fiecare ciclu elementar care trece printr-un vîrf x al grafului G utilizeazã douã muchii incidente cu x . Considerînd una din muchiile rãmase, ea aparține unui alt ciclu împreunã cu o altã muchie incidentã cu x ș.a.m.d. Deci $d(x)$ este par.

II.3.5. Se considerã graful obținut din G în modul următor : se adaugã un nou vîrf u împreunã cu muchiile $[u, x]$ și $[u, y]$

II.4.1. Se obțin trei arbori parțiali minimi. Unul dintre ei are muchiile $\{6, 7\}$, $\{1, 6\}$, $\{1, 4\}$, $\{1, 2\}$, $\{2, 3\}$ și $\{4, 5\}$.

II.4.2. a) \Rightarrow b) rezultã din propoziția II.4.2

Dacã are loc b), G fiind conex conține un arbore parțial A . Deoarece A are $n - 1$ muchii rezultã cã $G = A$, deci G nu conține cicluri, adicã are loc c). Dacă are loc c), sã presupunem prin reducere la absurd cã G nu este conex. Rezultã cã G are $p \geq 2$ componente conexe care conțin respectiv n_1, \dots, n_p vîrfuri. G fiind fãrã cicluri, fiecare componentã conexã este un arbore, deci G are $(n_1 - 1) + \dots + (n_p - 1) = n - p < n - 1$ muchii, contradicție. Deci G este arbore și c) \Rightarrow a).

II.4.3. G fiind conex conține un arbore parțial A . Suprimînd cîte un vîrf terminal al lui A și al subarborilor care se obțin în acest mod gãsîm un subarbore A_1 al lui A cu k vîrfuri. Subgraful lui G care are aceeași mulțime de vîrfuri cu A_1 satisface condițiile problemei.

II.4.4. Sã presupunem prin reducere la absurd cã existã un graf G cu n vîrfuri și cel puțin n muchii care nu conține nici un ciclu.

Rezultã cã G are $p \geq 1$ componente conexe, fiecare este un arbore și ele conțin respectiv n_1, \dots, n_p vîrfuri. Deci G are $(n_1 - 1) + \dots + (n_p - 1) = n - p \leq n - 1$ muchii, contradicție.

II.4.5. Sã presupunem cã prin eliminarea anumitor muchii din G am obținut un graf fãrã cicluri, cu p componente conexe care sînt arbori și conțin n_1, \dots, n_p vîrfuri. Acest graf are $(n_1 - 1) + \dots + (n_p - 1) = n - p \leq n - 1$ muchii. Deci din graful G trebuie eliminate cel puțin $m - (n - 1) = m - n + 1$ muchii, acest numãr fiind suficient pentru a obține un graf fãrã cicluri dacã se suprimã toate muchiile care nu aparțin unui arbore parțial al lui G .

II.4.6. Condiția este necesarã, deoarece $d_1 + \dots + d_n = 2m = 2(n - 1)$. Suficiența se demonstreazã prin inducție dupã n . Pentru $n = 2$ obținem un singur arbore cu $d_1 = d_2 = 1$. Presupunînd proprietatea adevãratã pentru n , fie numerele $d_1 \geq \dots \geq d_{n+1} \geq 1$ cu $d_1 + \dots + d_{n+1} = 2n$. Rezultã $d_{n+1} = 1$. Deci $d_1 + \dots + d_n = 2n - 1$. Obținem $d_1 \geq 2$, deoarece în caz contrar am avea $d_1 + \dots + d_n = n < 2n - 1$. Notînd $d'_1 = d_1 - 1$, rezultã $d'_1 + d_2 + \dots + d_n = 2n - 2$ și conform ipotezei de inducție existã un arbore A_1 cu gradele celor n vîrfuri egale respectiv cu $d_1 - 1, d_2, \dots, d_n$.

Arborele cãutat A se obține din A_1 considerînd un nou vîrf pe care îl legãm printr-o muchie d e vîrfului lui A_1 de grad $d_1 - 1$.

II.5.1. Dacă facem convenția ca operațiile de aceeași prioritate sã se efectueze de la stînga la dreapta, se obțin rezultatele: a) $** + xy + yx + zx$.

b) $+ - * 3 * \uparrow x 3 y * 5 * \uparrow x 2 \uparrow y 2 * 2 * x \uparrow y 3$.

c) $+ + / + xy + yz / + yz + zx / + zx + xy$

d) $- + + \uparrow a 3 \uparrow b 3 \uparrow c 3 * 3 * a * b c$.

e) $* - * 2 \uparrow x + * 3 y z a - * 14 \uparrow 3$.

f) $+ \uparrow a \uparrow b c / x + 2 / y z$.

II.5.2. Se utilizează inducția după $n \geq 2$. Orice expresie aritmetică cu n operanzi conține $n - 1$ operatori.

II.5.3. a. $5xyz - 4x^2y - 3z$;

b. $\frac{x}{y + \frac{x^2}{\dots}}$; c. x^{a+b+c} ;

d. $((x + 3a)(y - 1) + 2(x^2 + 5))$

II.5.4. Deoarece $2^3 < 10 < 2^4$ obținem un arbore binar complet cu 8 vîrfuri terminale pe nivelul 3 și 4 vîrfuri terminale pe nivelul 4.

II.5.5. Dacă alfabetul conține literele a_1, a_2, \dots, a_{26} în această ordine, deci putem scrie $a_1 < a_2 < \dots < a_{26}$, ordinea cuvintelor formate cu aceste litere într-un dicționar se definește astfel: cuvîntul $x = x_1 \dots x_n$ spunem că este mai mic decît cuvîntul $y = y_1 \dots y_m$ și el va fi scris în dicționar înaintea cuvîntului y dacă:

- x este un început propriu al lui y , adică $m > n$ și $x_1 = y_1, \dots, x_n = y_n$ sau
- există un indice $p \geq 1$ astfel încît $x_1 = y_1, \dots, x_{p-1} = y_{p-1}$ și $x_p < y_p$.

Această relație de ordine este o *relație de ordine totală*, adică pentru $x \neq y$ avem sau $x < y$ sau $y < x$.

II.5.6. Întîi se sortează cele 1 000 000 de numere, adică se obține un șir

$$a_1 \leq a_2 \leq \dots \leq a_{1\,000\,000}.$$

Apoi se parcurge secvențial lista acestor numere, mărind cu o unitate valoarea unui contor c de fiecare dată cînd întîlnim un nou număr, conform următorului algoritm:

- $c \leftarrow 1$;
- $i \leftarrow 1$;
- $i = 1\,000\,000$? Dacă da, ne oprim. Dacă nu, trecem la pasul următor.
- $a_{i+1} > a_i$? Dacă da, $c \leftarrow c + 1$ și mergem la 5. Dacă nu, mergem la 5.
- $i \leftarrow i + 1$ și se merge la 3.

La sfîrșitul aplicării algoritmului valoarea variabilei c va indica răspunsul la problemă.

II.5.7. Dacă comparăm toate cele C_{2001}^2 perechi de cuvinte binare trebuie să facem 8 002 000 comparații.

Pentru a reduce numărul comparațiilor, procedăm astfel: Se sortează întîi fișierul, astfel încît $a_1 < a_2 < \dots < a_{2001}$. Trebuie să verificăm condiția $a_i + a_j = c$, unde suma se face în baza 2 și $c = 11 \dots 1$. Ținînd seama că fișierul este sortat, putem proceda astfel:

- $i \leftarrow 1, j \leftarrow 4\,001$ și apoi se repetă pasul 2 pînă cînd $j \leq i$;
- Dacă $a_i + a_j = c$, se tipărește perechea $\{a_i, a_j\}$. Se stabilește $i \leftarrow i + 1, j \leftarrow j - 1$;
- Dacă $a_i + a_j < c$, se face $i \leftarrow i + 1$;
- Dacă $a_i + a_j > c$, se face $j \leftarrow j - 1$.

Se observă că la pasul 2 se fac cel mult $n - 1$ comparații cu c , dacă sînt n cuvinte în fișier. Deci în cazul nostru se fac cel mult 4 000 comparații. Pentru a sorta cele 4 001 cuvinte binare, considerate ca numere scrise în baza 2, mai sînt necesare cel mult 48 000 comparații. Rezultă un total de cel mult 52 000 comparații.

II.5.8. Se construiește un arbore binar complet de sortare cu vîrfurile terminale pe cel mult două nivele consecutive. Cei n jucători se asociază celor n vîrfuri terminale, fiecare nod lăta avînd asociat cîștigătorul meciului direct dintre cei doi filii săi. Cîștigătorul turneului va fi asociat rădăcinii arborelui. Pentru a găsi al II-lea cel mai bun jucător trebuie să joace între ei

învingător campionului. Am văzut că sînt suficiente $r = \lceil \log_2 n \rceil$ comparații la nivelele $r, r-1, \dots, 1$, deci $\lceil \log_2 n \rceil$ meciuri. Dacă $n = 2^r$, acest număr de meciuri se reduce la $r-1$, deoarece toate nodurile terminale sînt la nivelul r , deci trebuie făcute noi comparații la nivelele $r-1, r-2, \dots, 1$.

II.5.9. Algoritmul schimbă între ele oricare două numere vecine, dintre care cel mai mare se găsește înaintea celui mai mic. La fiecare etapă indicele i are proprietatea că înregistrările a_{i+1}, \dots, a_n se găsesc în poziția lor finală. Dacă $i = 0$, înseamnă că șirul de numere obținut verifică proprietatea $a_j \leq a_{j+1}$ pentru $j = 1, \dots, i-1$. Cum numerele a_{i+1}, \dots, a_n se găsesc în pozițiile finale, rezultă că șirul este sortat și ne oprim.

Dacă definim o *inversiune* ca fiind o pereche de numere $\{a_i, a_j\}$ cu $i < j$ și $a_i > a_j$, rezultă că șirul a_1, \dots, a_n care trebuie sortat prezintă cel mult C_n^2 inversiuni, în cazul cînd $a_1 > a_2 > \dots > a_n$. Șirul a_1, \dots, a_n este sortat dacă și numai dacă el prezintă zero inversiuni.

Dacă $a_j > a_{j+1}$, prin interschimbarea numerelor a_j și a_{j+1} între ele, șirul obținut are cu o inversiune mai puțin decît șirul de la care am plecat. Deci după un număr finit de interschimbări obținem un șir cu zero inversiuni, deci un șir sortat.

II.5.10. Se compară primele numere din cele două șiruri sortate, iar cel mai mic se trece în șirul final. Se repetă operația pentru șirurile obținute ș.a.m.d. Cînd unul din șiruri este epuizat, toate numerele rămase în celălalt se trec în șirul final, ele fiind cele mai mari elemente din cele două șiruri. Putem deci scrie:

1. Se stabilește $i \leftarrow 1, j \leftarrow 1, k \leftarrow 1$.
2. Dacă $a_i \leq b_j$, mergi la pasul 3, altfel treci la pasul 5.
3. Se fac atribuirile $c_k \leftarrow a_i, k \leftarrow k+1, i \leftarrow i+1$. Dacă $i \leq m$, mergi la 2. În caz contrar, treci la pasul următor.
4. Stabilește $(c_k, \dots, c_{m+n}) \leftarrow (b_j, \dots, b_n)$ și termină algoritmul.
5. Stabilește $c_k \leftarrow b_j, k \leftarrow k+1, j \leftarrow j+1$. Dacă $j \leq n$, mergi la 2, altfel treci la pasul următor.
6. Stabilește $(c_k, \dots, c_{m+n}) \leftarrow (a_i, \dots, a_m)$ și termină aplicarea algoritmului.

La pașii 4, respectiv 6 ajungem cînd unul din șiruri este parcurs în întregime, copîind în șirul final, ceea ce a mai rămas din celălalt șir.

La fiecare comparație de la pasul 2 transmitem în șirul final c_1, \dots, c_{m+n} cel puțin un număr, la pașii 3 sau 5. Deoarece ultimul număr este trecut în șirul final fără nici o comparație, rezultă că acest algoritm de interclasare necesită cel mult $m+n-1$ comparații.

Acest număr maxim de comparații este atins cînd de exemplu avem $\max(a_1, \dots, a_{m-1}) \leq b_1$ și $a_m > b_n$.

II.5.11. Algoritmul de căutare este următorul:

1. $i \leftarrow 1$.
2. $b = a_i$? Da: Stop. Căutarea a avut succes. Nu: Treci la pasul următor.
3. $b < a_i$? Da: Stop. Căutarea nu a avut succes. Nu: Treci la pasul următor.
4. $i = 4$? Da: Stop. Căutarea nu a avut succes și $b > a_4$.
Nu: $i \leftarrow i+1$ și se merge la pasul 2.

II.5.12. În cazul arborelui de căutare din figura II.48 numărul total de comparații în cazul unei căutări cu succes este egal cu 8.

Deci numărul mediu de comparații în cazul unei căutări cu succes este egal cu $\frac{8}{4} = 2$ comparații.

Dacă $b < a$, ajungem în nodul terminal numerotat cu 0 și facem 2 comparații: cu a , și cu a , etc.

Deci numărul total de comparații pentru căutările fără succes este egal cu 12, iar numărul mediu este egal cu $\frac{12}{5} = 2,4$ comparații

Pentru arborele de căutare din figura 11 49 obținem în mod analog numărul mediu de comparații pentru căutarea cu succes egal cu 2,5 comparații. Pentru căutarea fără succes numărul mediu de comparații este egal cu 2,8.

Rezultă că arborele de căutare din figura 11 48 este superior arborelui din figura 11 49 în ceea ce privește realizarea unui număr mediu cit mai mic de comparații

II.5.13. Se poate aplica următoarea idee: Ori de câte ori o înregistrare a fost localizată cu succes, ea este deplasată la începutul benzii, provocând translația celorlalte înregistrări, în aceeași ordine, către sfârșitul benzii.

După ce s-au făcut un număr de căutări cu succes ale înregistrărilor de pe bandă, înregistrările mai des căutate vor ocupa poziții mai apropiate de începutul benzii.

Cap. III

III.1.1. Se găsesc 4 drumuri care folosesc arcul (1, 3) și 4 drumuri care folosesc arcul (1, 2). Singurele circuite elementare sînt circuitele de lungime 2: (3, 4, 3) și (5, 6, 5)

III.1.2. Atît în suma gradelor de intrare cît și în suma gradelor de ieșire fiecare arc este numărat o dată și numai o dată, deoarece iese dintr-un vîrf și intră într-un singur vîrf

III.1.3. Oricare pereche de vîrfuri $\{x, y\}$ cu $x \neq y$ poate fi unită prin zero arce, prin arcul (x, y) , prin arcul (y, x) sau prin ambele arce (x, y) și (y, x) . Rezultă în total $4C_n^2 = 2^{n(n-1)}$ grafuri orientate cu n vîrfuri. Dacă graful este complet prima posibilitate este exclusă. Deci există $3C_n^2$ grafuri orientate și complete cu n vîrfuri.

III.1.4. Conform definiției, relația binară este relativă și simetrică. Pentru a arăta că este tranzitivă, se observă că dacă D_1 este un drum de la x la y și D_2 este un drum de la y la z , atunci există un drum D_3 de la x la z obținut din șirul vîrfurilor lui D_1 care se continuă cu șirul vîrfurilor din D_2 : $D_3 = (x_1, \dots, y, \dots, z)$.

În mod analog se arată că există un drum și de la z la x .

Pentru graful din figura III.1 componentele tare conexe sînt următoarele: $\{1, 2, 3\}$, $\{4\}$, $\{5\}$, $\{6, 7, 8\}$.

III.1.5. Dacă D nu este drum elementar, fie z primul vîrf întîlnit în sensul de la x la y care se repetă în șirul de vîrfuri care îl definește pe D . Vom suprima din D subșirul de vîrfuri cuprins între prima și ultima apariție a lui z în șirul D , păstrînd în D o singură apariție a lui z . Aplicăm același procedeu pentru drumul obținut, pînă la găsirea unui drum elementar de aceeași extremități.

III.1.6. Vom presupune prin reducere la absurd că graful-turneu G nu conține un drum elementar care trece prin toate vîrfurile grafului.

Fie $D = (x_1, x_2, \dots, x_p)$ un cel mai lung drum elementar al lui G . Va exista un vîrf y diferit de x_1, \dots, x_p . Graful fiind complet, y este legat prin arce cu vîrfurile x_1, \dots, x_p .

Dacă există arcul (y, x_1) obținem un drum mai lung ca D , ceea ce contrazice presupunerea făcută.

Deci există arcul (x_1, y) . Analog, dacă există arcul (x_p, y) , obținem un drum mai lung ca D . Deci există arcul (y, x_p) . Vor exista deci două vîrfuri consecutive x_k și x_{k+1} ale lui D pentru care arcele care le unesc cu y să aibă sensuri contrare și anume să existe arcele (x_k, y) și (y, x_{k+1}) . Am obținut iarăși un drum mai lung ca D și anume: $(x_1, x_2, \dots, x_k, y, x_{k+1}, \dots, x_p)$, ceea ce este contradictoriu.

Deci D conține toate vîrfurile grafului G .

III.1.7. Egalitatea 1) exprimă faptul că fiecare vîrf este incident cu exact $n - 1$ arc. Pentru a demonstra 2) se ține seama de problema III.1.2 și de faptul că orice graf-turneu are C_n^2 arce. Pentru a demonstra 3) se înmulțește 1) cu r_n , respectiv cu s , și se însumează egalitățile obținute, ținînd seama de 2).

III.1.8. Se va alege pentru x un vîrf pentru care $d^+(x)$ este maxim în G .

III.2.1. Se obțin două drumuri critice de lungime egală cu 16 și anume: (1, 2, 5, 8) și (1, 3, 7, 8). Deci data terminării lucrării este $t_8 = 16$.

III.2.2. Se obține graful programului din figura V.4. Drumul critic este (1, 2, 3, 5, 6). Intervalele de fluctuație au fost reprezentate în pătrățelele asociate vîrfurilor și marginile totale ale operațiilor au fost trecute în paranteze lângă arcele respective.

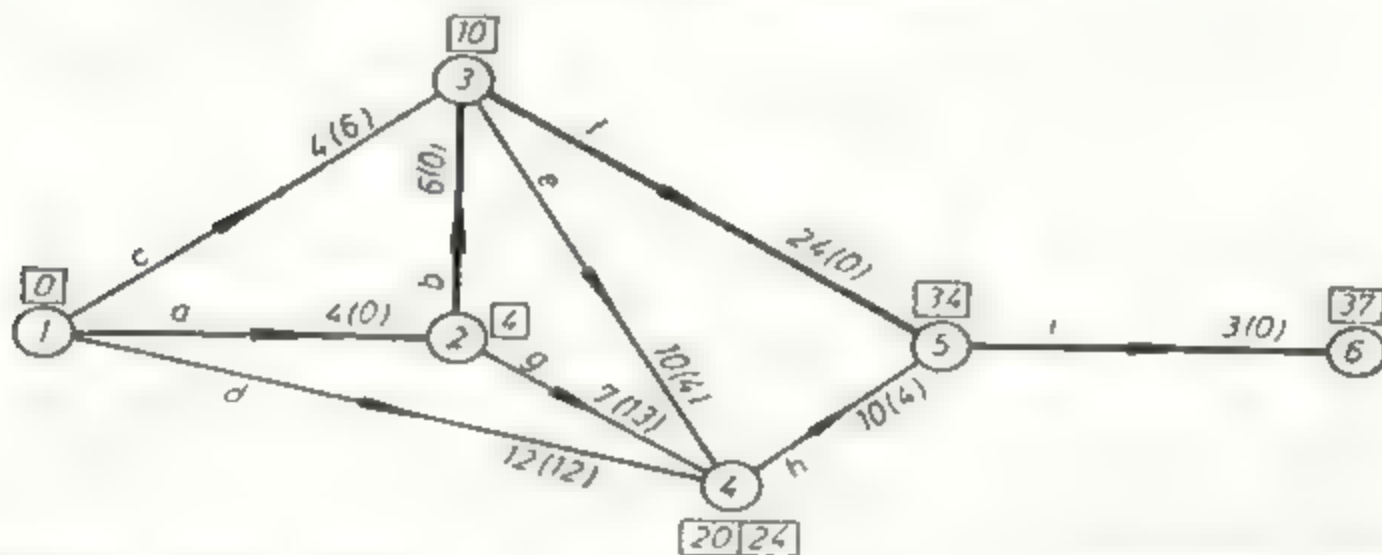


Fig. V.4

III.2.3. Să presupunem prin reducere la absurd că pentru orice vîrf x al grafului avem $p(x) \neq \emptyset$. Fie $D = (x_1, x_2, \dots, x_p)$ un drum elementar care are un număr maxim de arce în graful G . Rezultă că $p(x_1) \subset \{x_2, \dots, x_p\}$, deci există un indice k astfel încît $2 \leq k \leq p$ și $x_k = p(x_1)$. Am obținut un circuit elementar în graful G :

$$C = (x_1, x_2, \dots, x_k, x_1).$$

ceea ce contrazică ipoteza că G este fără circuite. În mod analog se demonstrează existența unui vîrf y cu proprietatea $s(y) = \emptyset$, considerînd extremitatea x_p a drumului D .

III.2.4. Deoarece fiecare din cele două arce disjunctive poate avea două orientări, rezultă în total 4 grafuri de activități. Dintre acestea, graful cu o lungime minimă a drumului critic, egală cu 16 unități, se obține dacă alegem arcul (3, 2) cu durata 6 și arcul (5, 4) cu durata 3. Deci trebuie ca evenimentul 3 să preceadă evenimentul 2 și evenimentul 5 să preceadă evenimentul 4.

III.3.1. $\max f_0 = 9$.

III.3.2. Se găsește fluxul maxim în rețeaua obținută din rețeaua dată prin înmulțirea capacităților tuturor arcelor cu $[2, 3, 11] = 66$. Apoi se revine la rețeaua inițială, împărțind toate componentele pe arce ale fluxului maxim astfel obținut prin 66. Se găsește $\max f_0 = \frac{69}{22}$ și o tăietură de capacitate minimă are suportul $A = \{2, 3, 4, 5, b\}$.

III.3.3. Rețeaua de transport asociată problemei are vîrfurile $x_1, x_2, x_3, y_1, y_2, y_3$. Intrarea a și ieșirea b . Arcele (a, x_i) au o capacitate egală cu a_i și arcele (y_i, b) au o capacitate egală cu b_i , pentru $i = 1, 2, 3$.

Arcelor de forma (x_i, y_j) le vom asocia o capacitate egală cu cantitatea de produse care pot fi transportate din x_i în y_j . Un flux compatibil cu capacitățile arcelor reprezintă un plan de transport posibil. Fluxul f_0 va reprezenta cantitatea totală de produse transportate din centrele x_i în centrele y_j . Aplicînd algoritmul lui Ford Fulkerson se găsește un flux maxim ale cărui componente pe arcele (x_i, y_j) ne dau următorul plan optim de transport, care utilizează tot disponibilul de la centrele x_1, x_2, x_3 :

	y_1	y_2	y_3
x_1	12	2	46
x_2	14	0	10
x_3	7	17	12

Cantitatea de produse transportate este egală cu $f_0 = 33 + 19 + 68 = 120$ unități. Soluția problemei nu este unică.

III.3.4. Deoarece $a \notin A$ și $b \in A$, rezultă că drumul D va conține cel puțin o pereche de virfuri vecine x_i și x_{i+1} , astfel încît $x_i \notin A$ și $x_{i+1} \in A$. În acest caz arcul $(x_i, x_{i+1}) \in \omega^-(A)$.

III.3.5. Graful din figura III 15 este o rețea de transport, unde costurile de instalare a posturilor de control devin capacități ale arcelor.

În această rețea trebuie să determinăm o mulțime de arce astfel încît orice drum de la a la b să conțină cel puțin un arc din această mulțime, iar suma capacităților acestor arce să fie minimă.

Această mulțime de arce formează o tăietură de capacitate minimă

Aplicînd algoritmul lui Ford-Fulkerson găsim o tăietură de capacitate minimă, de suport $A = \{4, b\}$, deci $\omega^-(A) = \{(1, 4), (5, 4), (5, b), (6, b)\}$ și $c(\omega^-(A)) = 2 + 2 + 6 + 5 = 15$ unități. Rezultă că posturile de control care supraveghează toate drumurile din a în b trebuie amplasate pe arcele din mulțimea $\omega^-(A)$, cu un cost total de instalare de 15 unități.

III.3.6. Se găsesc activitățile critice ale grafului din figura III.16, care sînt $(1, 2), (2, 3), (3, 6), (3, 4), (4, 5), (5, 6), (5, 7)$ și $(6, 7)$. Aceste arce formează graful parțial din figura V.5.

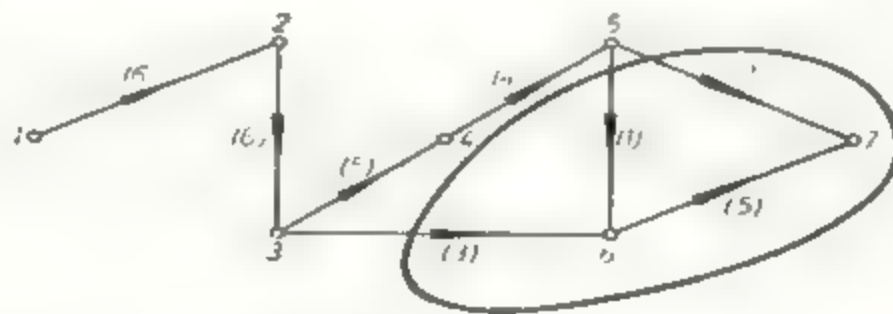


Fig. V.5

Asociînd arcelor din figura V.5 costurile din tabelul dat, se obține o rețea de transport cu intrarea 1 și ieșirea 7, pentru care aceste costuri pot fi interpretate drept capacități ale arcelor.

Problema revine la a determina o mulțime de arce, astfel încît orice drum de la 1 la 7 în graful din figura V.5 să conțină cel puțin un arc din această mulțime, iar suma capacităților acestor arce să fie minimă. Conform discuției de la problema precedentă, trebuie să găsim o

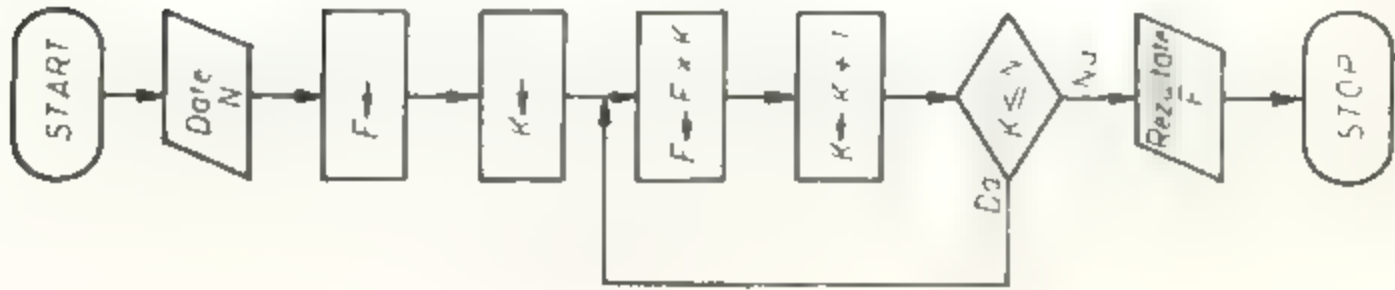


Fig V.6

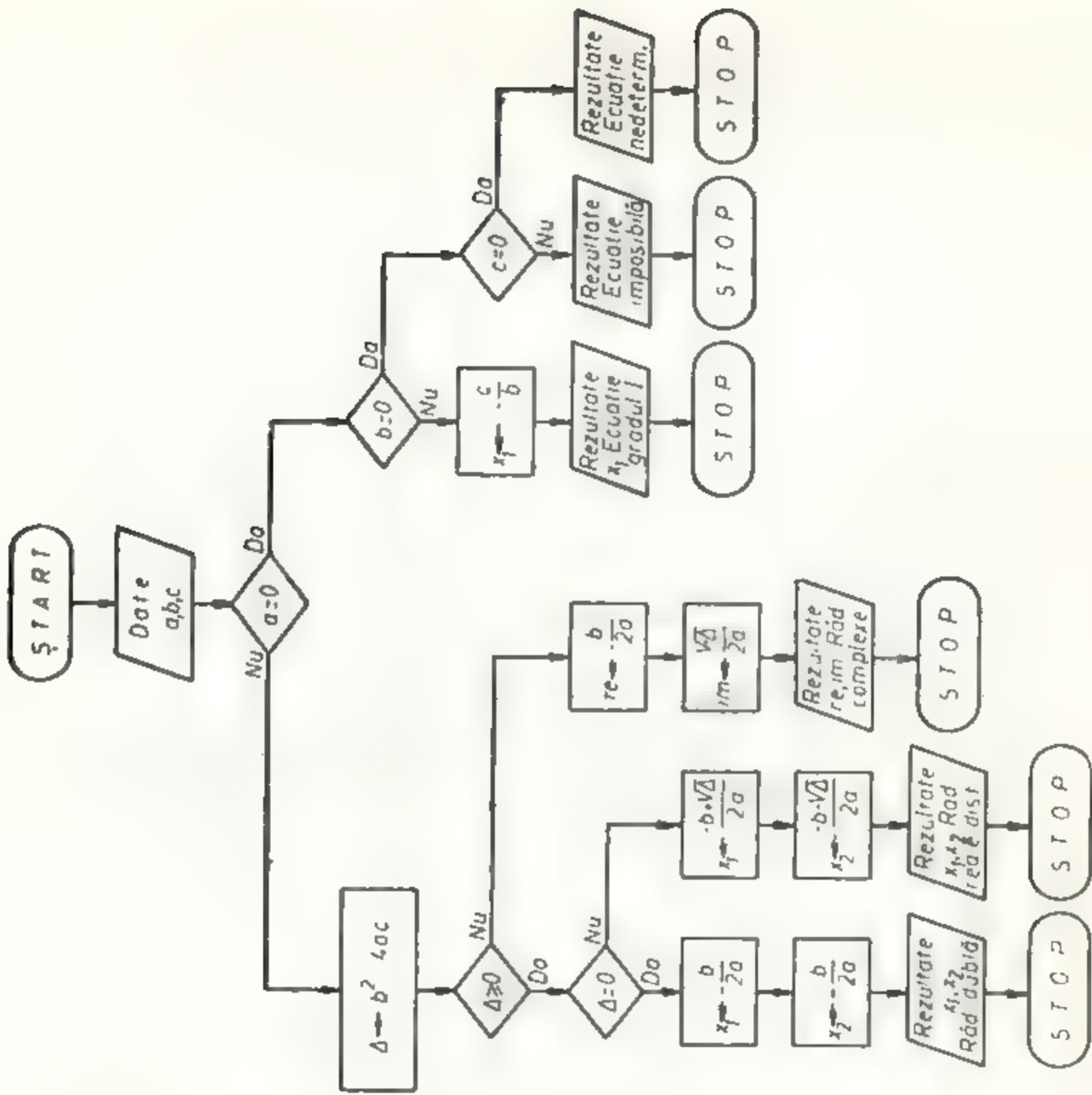


Fig V.7

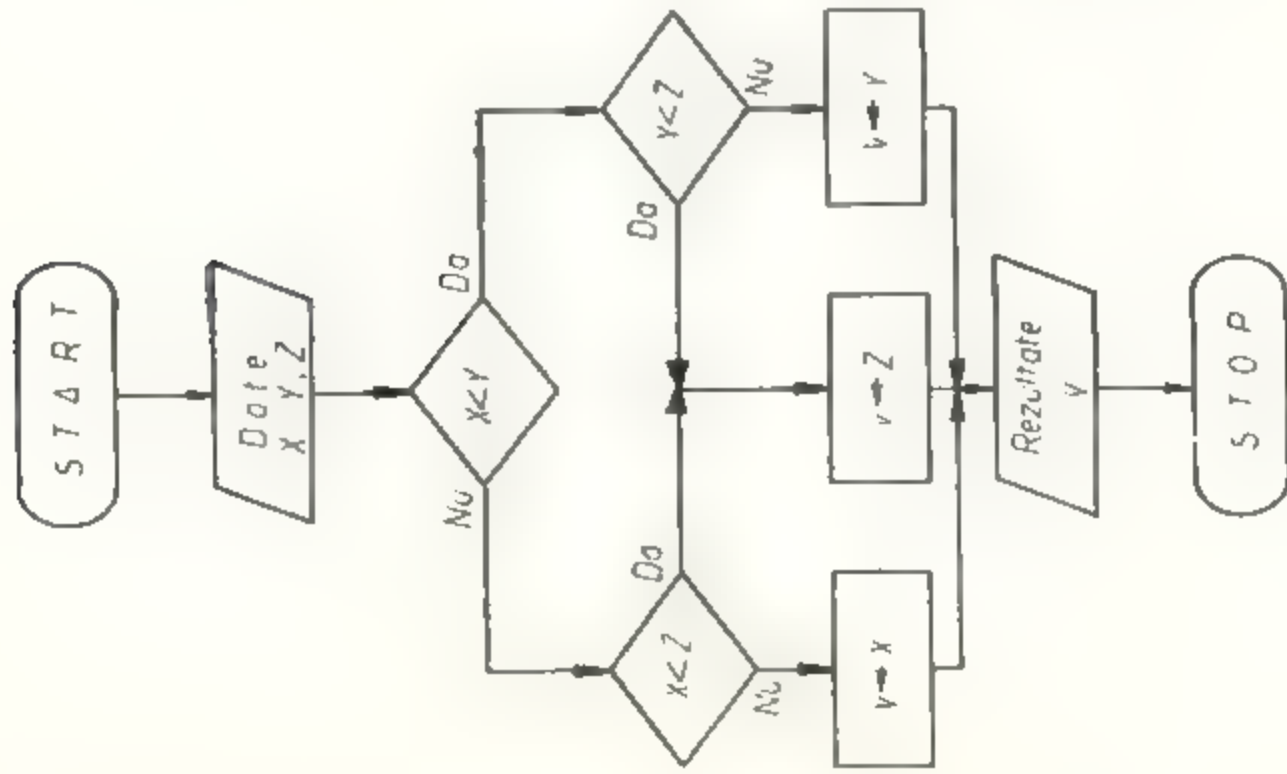


Fig. V.8

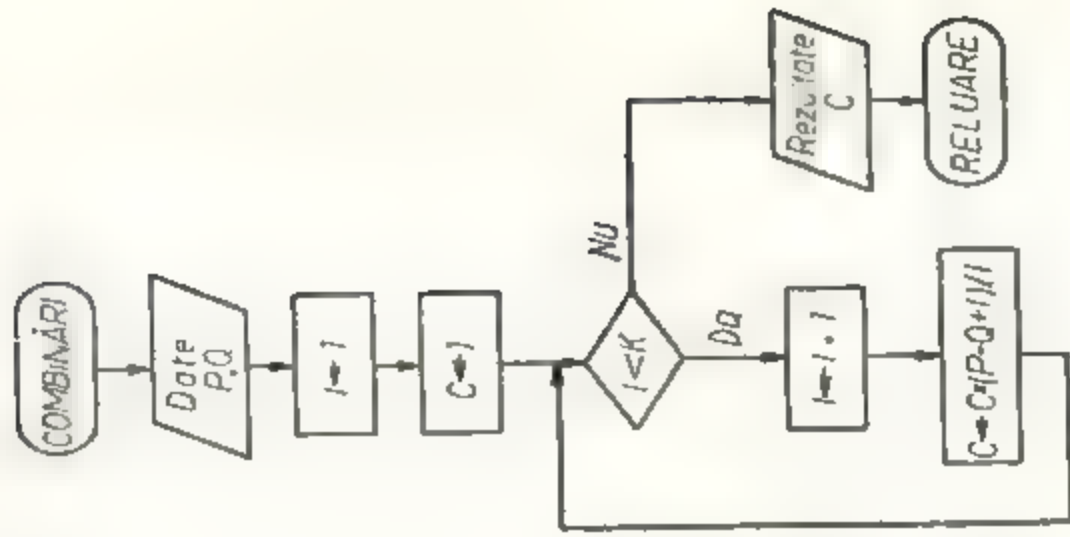
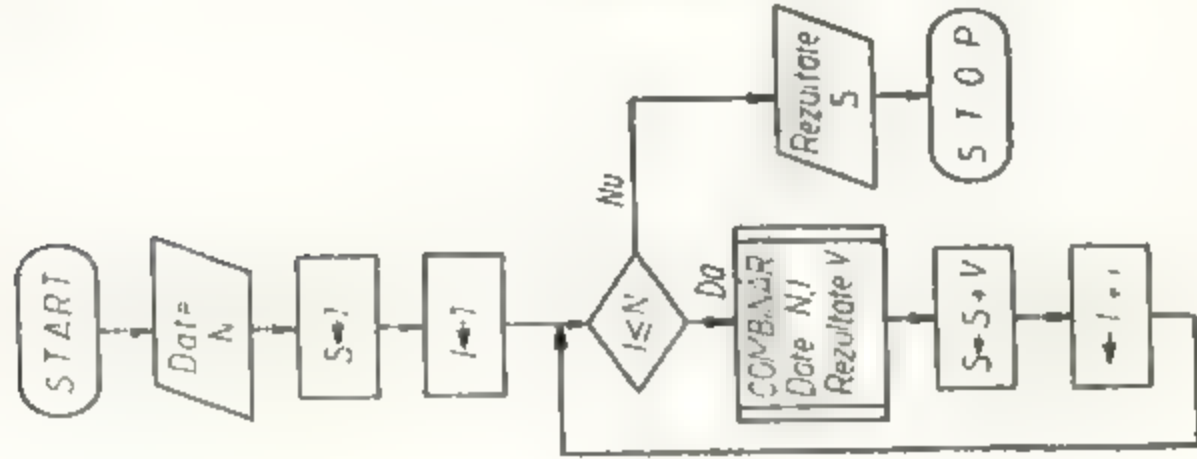


Fig. V.9

tăietură de capacitate minimă în rețeaua de transport din figura V.5. Această tăietură este formată din arcele (5, 7), (5, 6) și (3, 6), de capacitate $1 + 1 + 3 = 5$.

Rezultă că costul minim pentru scurtarea duratei întregii lucrări cu o unitate este egal cu 5 și el se realizează dacă scurtăm cu cîte o unitate duratele operațiilor critice (3, 6), (5, 6) și (5, 7).

Cap. IV

IV.1.5. 1. Nu, deoarece M nu se anulează și algoritmul nu se termină. 2. 225. 4. Pas 1. $v \leftarrow a_4$; Pas 2. $p \leftarrow x_0$; Pas 3. $v \leftarrow v + p \times a_1$; Pas 4. $p \leftarrow p \times x_0$; Pas 5. $v \leftarrow v + p \times a_2$; Pas 6. $p \leftarrow p \times x_0$; Pas 7. $v \leftarrow v + p \times a_1$; Pas 8. $p \leftarrow p \times x_0$; Pas 9. $v \leftarrow v + p \times a_2$; Pas 10. Term. alg. 5. Pas 1. Dacă $X < Y$ urmează pas 5; Pas 2. Dacă $X < Z$ urmează pas 4; Pas 3. $V \leftarrow X$ urmează pas 7; Pas 4. $V \leftarrow Z$ urmează pas 7; Pas 5. Dacă $Y < Z$ urmează pas 4; Pas 6. $V \leftarrow Y$; Pas 7. Term. alg. 6. Pas 1. $a_1 \leftarrow a$; Pas 2. $b_1 \leftarrow a_1 \times x_0 + b$; Pas 3. $c_1 \leftarrow b_1 \times x_0 + c$; Pas 4. $r \leftarrow c_1 \times x_0 + d$; Pas 5. Term. alg. 7. Pas 1. $S \leftarrow 0$; Pas 2. $I \leftarrow 0$; Pas 3. $T \leftarrow a$; Pas 4. $S \leftarrow S + T$; Pas 5. $T \leftarrow T \times q$; Pas 6. $I \leftarrow I + 1$; Pas 7. Dacă $I \leq n$

urmează pas 4; Pas 8. Term. alg. 8. Se folosește formula $C_k^k = \frac{n}{k} C_{k-1}^{k-1}$; Pas 1. $I \leftarrow 0$;

Pas 2. $C \leftarrow 1$; Pas 3. Dacă $I = k$ urmează pas 6; Pas 4. $C \leftarrow C \times (n - I)/(k - I)$ urmează pas 3; Pas 5. $I \leftarrow I + 1$; Pas 6. Term. alg.

IV.2.2. 1. (vezi fig. V.6). 2. (vezi fig. V.7); 3. 584; 133. 4. (vezi fig. V.8). 5. Pas 1. $V \leftarrow 1$; Pas 2. $I \leftarrow 1$; Pas 3. Dacă $I < N$ urmează pas 5; Pas 4. Term. alg.; Pas 5. $I \leftarrow I + 1$; Pas 6. $V \leftarrow 1/2(V + b/V)$ urmează pas 3. 6. a) Pas 1. $S \leftarrow 1$; Pas 2. $I \leftarrow 1$; Pas 3. $K \leftarrow 1$; Pas 4. Dacă $I \leq N$ urmează pas 8; Pas 5. $I \leftarrow I + 1$; Pas 6. $K \leftarrow K + 2$; Pas 7. $S \leftarrow S + K^2$ urmează pas 4; Pas 8. Term. alg. b) Pas 1. $S \leftarrow 0$; Pas 2. $K \leftarrow 1$; Pas 3. $T \leftarrow 1$; Pas 4. Dacă $K > N$ urmează pas 8; Pas 5. $S \leftarrow S + T$; Pas 6. $K \leftarrow K + 1$; Pas 7. $T \leftarrow T \times K$ urmează pas 4. Pas 8. Term. alg. c) (vezi fig. V.9).

Cuprins

I. ALGEBRĂ BOOLEANĂ	3
1.1. Noțiunea de algebră booleană	3
1.1.1. Lattice	3
1.1.2. Lattice distributive și complementate	6
1.1.3. Exerciții	9
1.2. Funcții booleene	10
1.2.1. Expresii booleene	10
1.2.2. Funcții booleene	12
1.2.3. Reprezentarea funcțiilor booleene	13
1.2.4. Exerciții	16
1.3. Forme canonice ale funcțiilor booleene	18
1.3.1. Forme normale ale funcțiilor booleene	18
1.3.2. Forme canonice ale funcțiilor booleene	19
1.3.3. Exerciții	22
1.4. Simplificarea funcțiilor booleene	23
1.4.1. Simplificarea prin calcul boolean	23
1.4.2. Simplificarea prin diagrame	24
1.4.3. Exerciții	29
1.5. Realizarea fizică a funcțiilor booleene	31
1.5.1. Circuite cu contacte	31
1.5.2. Circuite logice simple	33
1.5.3. Proiectarea circuitelor logice	38
1.5.4. Circuite logice complexe	41
1.5.5. Realizarea practică a circuitelor logice	45
1.5.6. Exerciții	46
II. GRAFURI NEORIENTATE	49
11.1. Noțiuni de bază	49
Probleme	54
11.2. Grafuri hamiltoniene	56
Probleme	58
11.3. Grafuri euleriene	59
Probleme	61
11.4. Arbori	62
Probleme	66
11.5. Arbori binari și aplicații	67
11.5.1. Notăția fără paranteze a unei expresii aritmetice	68
11.5.2. Arbori de sortare	70
11.5.3. Algoritm de căutare binară	74
Probleme	78

III. GRAFURI ORIENTATE	80
III.1. Noțiuni de bază	80
Probleme	82
III.2. Metoda drumului critic	83
III.2.1. Drumul critic într-un graf de activități	83
III.2.2. Evenimente critice și intervale de fluctuație	85
III.2.3. Determinarea drumului critic	88
Probleme	91
III.3. Flux maxim într-o rețea de transport	92
Probleme	99
IV. ALGORITMI ȘI METODE DE REPREZENTARE	102
IV.1. Noțiunea de algoritm	102
IV.1.1. Definiția algoritmului	102
IV.1.2. Proprietățile algoritmilor	104
IV.1.3. Structura algoritmilor	104
IV.1.4. Clasificarea algoritmilor	107
IV.1.5. Exerciții	108
IV.2. Metode de reprezentare a algoritmilor	109
IV.2.1. Scheme logice	109
IV.2.2. Exerciții	112
Indicații și răspunsuri	114

